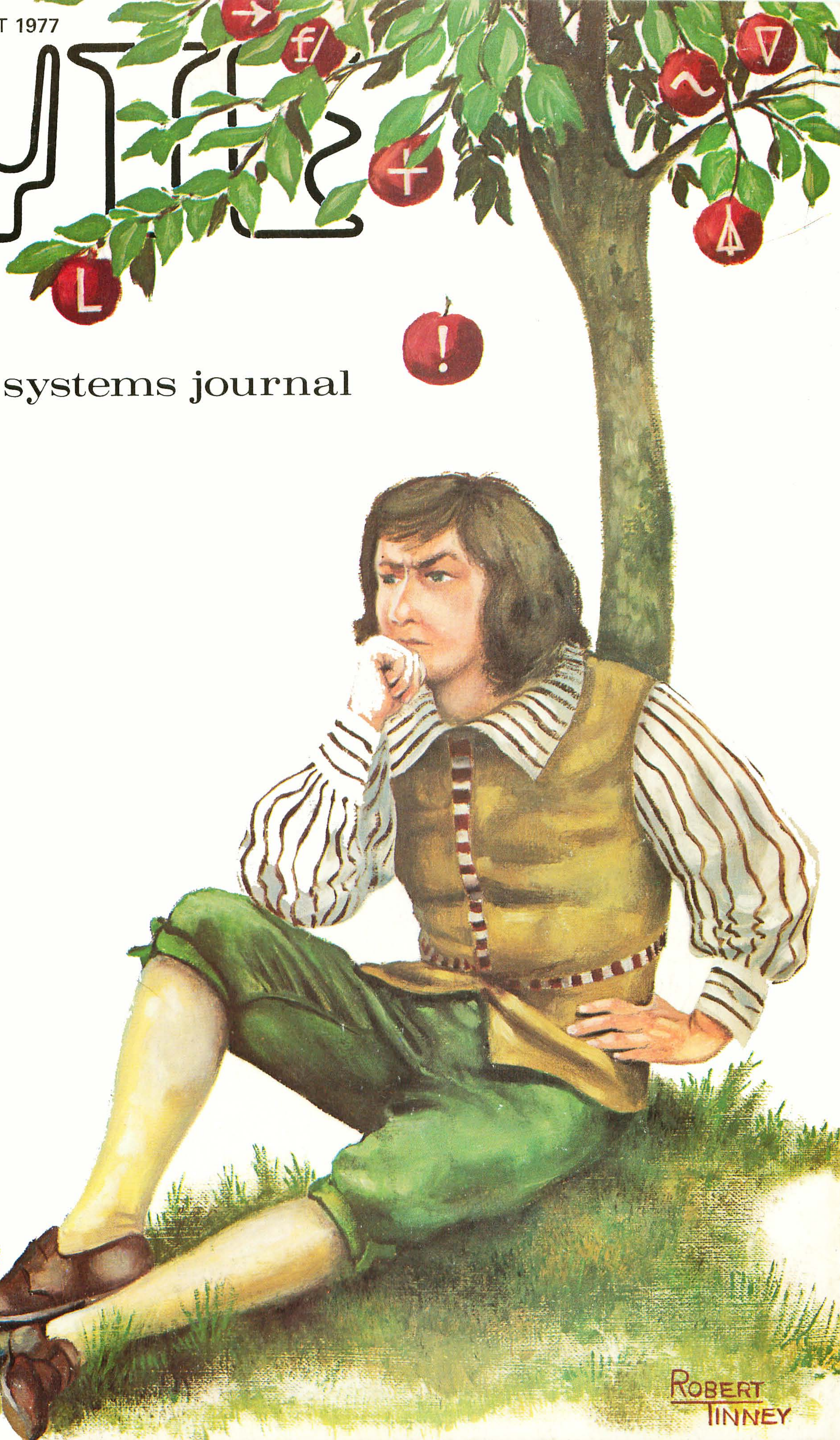
$1.50 in USA AUGUST 1977

BYTE

VOLUME 2, Number 8

the small systems journal



SWTPC announces first dual minifloppy kit under $1,000



Now SWTPC offers complete best-buy computer system with $995 dual minifloppy, $500 video terminal/monitor, $395 4K computer.



$995 MF-68 Dual Minifloppy
You need dual drives to get full benefits from a minifloppy. So we waited to offer a floppy until we could give you a dependable dual system at the right price.

The MF-68 is a complete top-quality minifloppy for your SWTPC Computer. The kit has controller, chassis, cover, power supply, cables, assembly instructions, two highly reliable Shugart drives, and a diskette with the Floppy Disk Operating System (FDOS) and disk BASIC. (A floppy is no better than its operating system, and the MF-68 has one of the best available.) An optional $850 MF-6X kit expands the system to four drives.



$500 Terminal/Monitor
The CT-64 terminal kit offers these premium features: 64-character lines, upper/lower case letters, switchable control character printing, word highlighting, full cursor control, 110-1200 Baud serial interface, and many others. Separately the CT-64 is $325, the 12 MHz CT-VM monitor $175.



$395 4K 6800 Computer
The SWTPC 6800 comes complete with 4K memory, serial interface, power supply, chassis, famous Motorola MIKBUG® mini-operating system in read-only memory (ROM), and the most complete documentation with any computer kit. Our growing software library includes 4K and 8K BASIC (cassettes $4.95 and $9.95; paper tape $10.00 and $20.00). Extra memory, $100/4K or $250/8K.

Other SWTPC peripherals include $250 PR-40 Alphanumeric Line Printer (40 characters/line, 5 x 7 dot matrix, 75 line/minute speed, compatible with our 6800 computer and MITS/IMSAI); $79.50 AC-30 Cassette Interface System (writes/reads Kansas City standard tapes, controls two recorders, usable with other computers); and other peripherals now and to come.

Circle 101 on inquiry card.

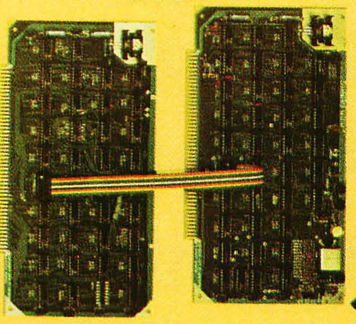
Enclosed is:

_____ $1,990 for the full system shown above (MF-68 Minifloppy, CT-64 Terminal with CT-VM Monitor).
_____ $995 for the Dual Minifloppy
_____ $325 for the CT-64 Terminal
_____ $175 for the CT-VM Monitor
_____ $395 for the 4K 6800 Computer
_____ $250 for the PR-40 Line Printer
_____ $79.50 for AC-30 Cassette Inferface
_____ Additional 4K memory boards at $100
_____ Additional 8K memory boards at $250
_____ Or BAC # __________ Exp. Date_____
_____ Or MC # __________ Exp. Date_____

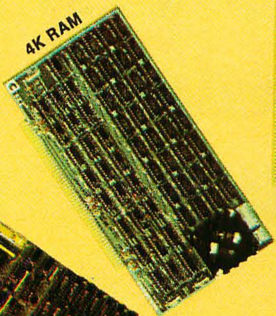
Name__________________ Address__________________
City__________________ State ________ Zip______

Southwest Technical Products Corp.
219 W. Rhapsody, San Antonio, Texas 78216
London: Southwest Technical Products Co., Ltd.
Tokyo: Southwest Technical Products Corp./Japan

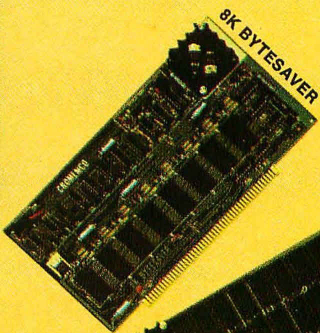
TV DAZZLER



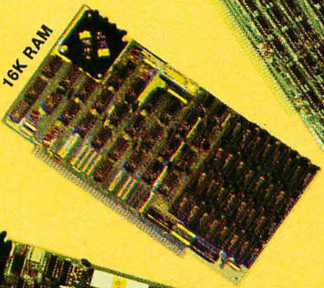
4K RAM



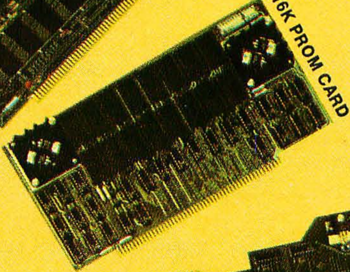
8K BYTESAVER



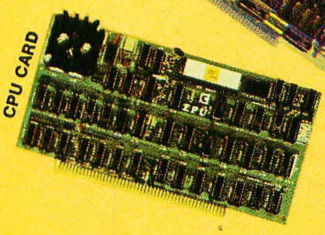
16K RAM



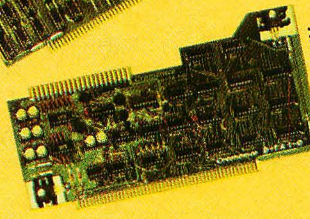
16K PROM CARD



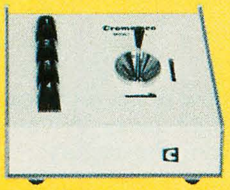
CPU CARD



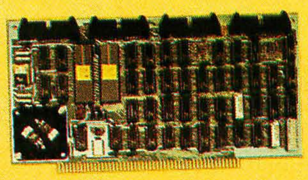
ANALOG I/O



JOYSTICK CONSOLE



TU-ART



To make your computer more useful– a wide choice of memory, I/O, CPU

Your computer's usefulness depends on the capability of its CPU, memories, and I/O interfaces, right?

So here's a broad line of truly useful computer products that lets you do interesting things with your Cromemco Z-1 and Z-2 computers. And with your S-100-compatible Altairs and IMSAIs, too.

CPU

- Z-80 MICROPROCESSOR CARD. The most advanced μP card available. Forms the heart of our Z-1 and Z-2 systems. Also a direct replacement for Altair/IMSAI CPUs. Has 4-MHz clock rate and the power of the Z-80 μP chip. Kit (Model ZPU-K): $295. Assembled (Model ZPU-W): $395.

MEMORIES

- 16K RAM. The fastest available. Also has bank-select feature. Kit (Model 16KZ-K): $495. Assembled (Model 16KZ-W): $795.
- 4K RAM. Bank-select allows expansion to 8 banks of 64K bytes each. Kit (Model 4KZ-K): $195. Assembled (Model 4KZ-W): $295.
- THE BYTESAVER — an 8K capacity PROM card *with integral programmer*. Uses high-speed 2708 erasable PROMs. A must for all computers. Will load 8K BASIC into RAM in less than a second. Kit (Model BSK-0): $145. Assembled (Model BSW-0): $245.
- 16K CAPACITY PROM CARD. Capacity for up to 16K of high-speed 2708 erasable PROM. Kit (Model 16KPR-K): $145. Assembled (Model 16KPR-W): $245.

I/O INTERFACES

- FAST 7-CHANNEL DIGITAL-ANALOG I/O. Extremely useful board with 7 A/D channels and 7 D/A channels. Also one 8-bit parallel I/O channel. Kit (Model D + 7A-K): $145. Assembled (Model D + 7A-W): $245.
- TV DAZZLER. Color graphics interface. Lets you use color TV as full-color graphics terminal. Kit (Model CGI-K): $215. Assembled (Model CGI-W): $350.
- DIGITAL INTERFACE (OUR NEW TU-ART). Interfaces with teletype, CRT terminals, line printers, etc. Has not one but *two serial I/O ports* and *two 8-bit parallel I/O ports* as well as 10 on-board interval timers. Kit (Model TRT-K): $195. Assembled (Model TRT-W): $295.
- JOYSTICK. A console that lets you input physical position data with above Model D + 7 A/D card. For games, process control, etc. Contains speaker for sound effects. Kit (Model JS-1-K): $65. Assembled (Model JS-1-W): $95.

PROFESSIONAL QUALITY

You get first-class quality with Cromemco.

Here are actual quotes from articles by independent experts: "The Cromemco boards are absolutely beautiful" . . . "The BYTESAVER is tremendous" . . . "Construction of Cromemco I/O and joystick are outstanding" . . . "Cromemco peripherals ran with no trouble whatsoever."

Everyone agrees. Cromemco is tops.

STORES/MAIL

So count on Cromemco. Look into these Cromemco products at your store. Or order by mail from the factory.

We wish you pleasure and success with your computer.

Cromemco
incorporated
Specialists in computers and peripherals
2432 CHARLESTON RD., MOUNTAIN VIEW, CA 94043 • (415) 964-7400

Circle 102 on inquiry card.

Meet the most powerful μC system available for dedicated work. Yet it's only $595.*

Here's the muscle you've been telling us you wanted: a powerful Cromemco microcomputer in a style and price range ideal for your dedicated computer jobs—ideal for industrial, business, instrumentation and similar applications.

It's the new Cromemco Z-2 Computer System. Here's some of what you get in the Z-2 for only $595:

- The industry's fastest μP board (Cromemco's highly regarded 4 MHz, 250-nanosecond cycle time board).
- The power and convenience of the well-known Z-80 μP.
- A power supply you won't believe (+8V @ 30A, +18V and −18V @ 15A — ample power for additional peripherals such as floppy disk drives).
- A full-length *shielded* motherboard with 21 card slots.
- Power-on-jump circuitry to begin automatic program execution when power is turned on.
- S-100 bus.
- Standard rack-mount style construction.
- All-metal chassis and dust case.
- 110- or 220-volt operation.

DEDICATED APPLICATIONS

The new Z-2 is specifically designed as a powerful but economical dedicated computer for systems work. Notice that the front panel is entirely free of controls or switches of any kind. That makes the Z-2 virtually tamper-proof. No accidental program changes or surprise memory erasures.

FASTEST, MOST POWERFUL μC

Cromemco's microcomputers are the fastest and most powerful available. They use the Z-80 microprocessor which is widely regarded as the standard of the future. So you're in the technical fore with the Z-2.

BROAD SOFTWARE/PERIPHERALS SUPPORT

Since the Z-2 uses the Z-80, your present 8080 software can be used with the Z-2. Also, Cromemco offers broad software support including a monitor, assembler, and a BASIC interpreter.

The Z-2 uses the S-100 bus which is supported by the peripherals of dozens of manufacturers. Naturally, all Cromemco peripherals such as our 7-channel A/D and D/A converter, our well-known BYTESAVER with its built-in PROM programmer, our color graphics interface, etc., will also plug into the S-100 bus.

LOW, LOW PRICE

You'll be impressed with the Z-2's low price, technical excellence and quality. So see it right away at your computer store—or order directly from the factory.

Z-2 COMPUTER SYSTEM KIT (MODEL Z-2K) (includes 4 MHz μP card, full-length 21-card-slot motherboard, power supply, one card socket and card-guide set, and front panel; for rack mounting) . . . . $595.

Z-2 COMPUTER SYSTEM ASSEMBLED (MODEL Z-2W) (includes the above as well as all 21 sockets and card guides and a cooling fan; for rack mounting) . . . $995.



Shown with optional bench cabinet

***kit price**

Cromemco
I n c o r p o r a t e d
Specialists in computers and peripherals
2432 CHARLESTON RD., MOUNTAIN VIEW, CA 94043 • (415) 964-7400

Circle 102 on inquiry card.

In the Queue

Foreground

Background

Nucleus

Cover: NEWTON UNDER THE APL TREE by Robert Tinney

BYTE is published monthly by BYTE Publications Inc, 70 Main St, Peterborough NH 03458. Address all mail except subscriptions to above address; phone (603) 924-7217. Address all editorial correspondence to the editor at the above address. Unacceptable manuscripts will be returned if accompanied by sufficient first class postage. Not responsible for lost manuscripts or photos. Opinions expressed by the authors are not necessarily those of BYTE. Address all subscriptions, change of address, Form 3579, and fulfillment complaints to BYTE Subscriptions, PO Box 361, Arlington MA 02174; phone (617) 646-4329.

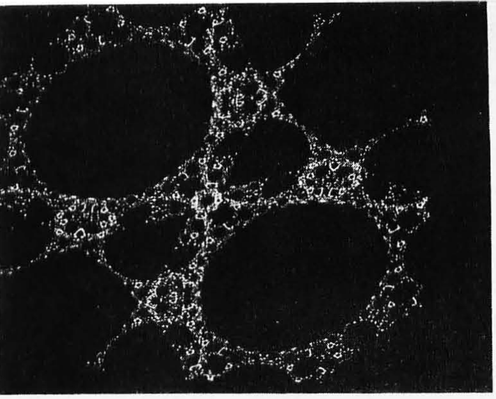
Second class postage paid at Peterborough NH 03458 and at additional mailing offices—USPS Publication No. 102410.

Subscriptions are $12 for one year, $22 for two years, and $32 for three years in the USA and its possessions. Add $5.50 per year for subscriptions to Canada and Mexico. $25 for a one year subscription by surface mail worldwide. Air delivery to selected areas at additional rates available upon request. $25 for a one year subscription by air delivery to Europe, or DM 60,— for a one year subscription through our European distributor in Darmstadt, West Germany.

Single copy price is $1.50 in the USA and its possessions, $2 in Canada and Mexico, and $3 elsewhere, or DM 7,— through our European distributor in Darmstadt, West Germany.

Foreign subscriptions and sales should be remitted in United States funds. Printed in United States of America.

BYTE
Volume 2
Number 8
AUGUST 1977



page 70

In This BYTE

One of the most exciting aspects of experimentation with computers is using the computer to make objects move. Distilling this excitement to its simplest form, Leon Sweer, Thomas Dwyer and Margot Critchfield at the University of Pittsburgh's Project Solo have designed a "one-dimensional plotter" which is used as the output display for the familiar Lunar Lander simulation game. Read **Controlling Small DC Motors with Analog Signals.**

The small computer field is expanding fast. Growing with it is a need for more microprocessor oriented courses in our universities. In his article, **A Microprocessor Course**, Mark E Fohl explains how he helped initiate a microprocessor course at Franklin University in Columbus OH and details some of the difficulties instructors might expect to run into.

Read Joe Weisbecker's **COSMAC VIP, the RCA Fun Machine**, a personal description of his new COSMAC VIP kit. This article contains some background information on one man's design philosophy and its result in a new product from RCA.

The lead article of our special section on APL (A Programming Language) was written, appropriately, by the man who developed it at IBM during the early 1960s: Dr Kenneth E Iverson. In **Understanding APL** Dr Iverson analyzes some basic APL operators through a series of questions designed to test your powers of induction. The article concludes with some challenging APL "thought experiments" for your decoding pleasure.

An excellent way to become more familiar with APL is to read David D Keefe's **Here's APL in Action**! David has written an APL version of the popular Lunar Lander game along with a detailed description of the entire program to help the reader decipher those high information density APL statements.

Have you ever wanted to own a truly powerful computer? APL may be an answer to this quest for an ultimate language when it is added to an otherwise mundane computer. Mike Wimble's **An APL Interpreter for Microcomputers** can help you implement APL if you have a flair for do it yourself software. With a little effort you can use any of the popular processors to produce your own APL interpreter from Mike's design.

Have you always wanted to have an elegant looking computer art program for graphic display, but didn't want to spend the time required to program it? Try **Serendipitous Circles**! Authors Galway and Anderson have discovered a simple algorithm to create beautiful and unusual moving patterns in no time at all.

The ability to handle and manipulate text is a very useful function. It allows programs to be changed very quickly without having to manipulate large amounts of data (and it is certainly better than an IBM correcting Selectric typewriter for writing and manipulating text). In his article, **Editorializing with Your Computer**, Gary McGath describes some of the properties of simple text editing programs.

What makes APL such a heavily discussed topic among small system enthusiasts? How does it differ from BASIC, and why should you use it at all? Allen Atwood discusses all of these topics and more in **Why People Get Hooked on APL**. Find out what you've been missing by programming in BASIC.

Often it is necessary to determine the data that is present on a particular port or set of lines of a computer for purposes such as debugging. The usual indicator bit pattern consisting of individual lamps can be very confusing and lead to errors if read quickly. R R Burns offers a solution to this problem. In his article, **An 8 Digit Hexadecimal Readout**, he describes a test rig that can monitor 32 lines in groups of four bits.

In this issue, Ralph Tenny describes some of the tricks (some call them kluges) which can be done with pulse waveform edges, Schmitt triggers, resistors and capacitors. Turn to **Look What You Can Do . . . with an Edge as a Cue** to understand some of the design practices you may occasionally find in a processor.

In two earlier articles in BYTE W Douglas Maurer described an algorithm for processing algebraic expressions when implementing compilers or interpreters. In his latest contribution to BYTE's ongoing flow of tutorials on computer science problems and techniques, Dr Maurer presents an ingenious extension of the Bauer-Samelson algorithm which will enable the do-it-yourself compiler writer to create efficient object code for expressions involving logical operators and comparisons of operands.

Many people use their computers as number crunchers and want quick, accurate algorithms for determining some fairly involved mathematical functions. Typical examples are trigonometric functions, hyperbolic functions and exponentiation. In his article, **Simple Algorithms for Calculating Elementary Functions**, John Rheinstein details several algorithms to calculate these functions quickly.

How do you get rid of bugs? Set out a few traps using techniques outlined in John M Schulein's article on **A Trapping Technique for the 8080.**

Editorial

Comments on the Acquisition of Knowledge

By Carl Helmers

I am sometimes asked by persons new to the field, "What is the best way to approach the problem of learning enough about computers and their intricacies to allow the neophyte to accomplish useful tasks?" Some thoughts on this subject, with an aside on simplicity and system design, provide the theme for this essay on the philosophy of personal computing usage and experimentation. My point of departure for this thought journey is an old but useful princle adapted to a modern context.

The Principle of Simplicity

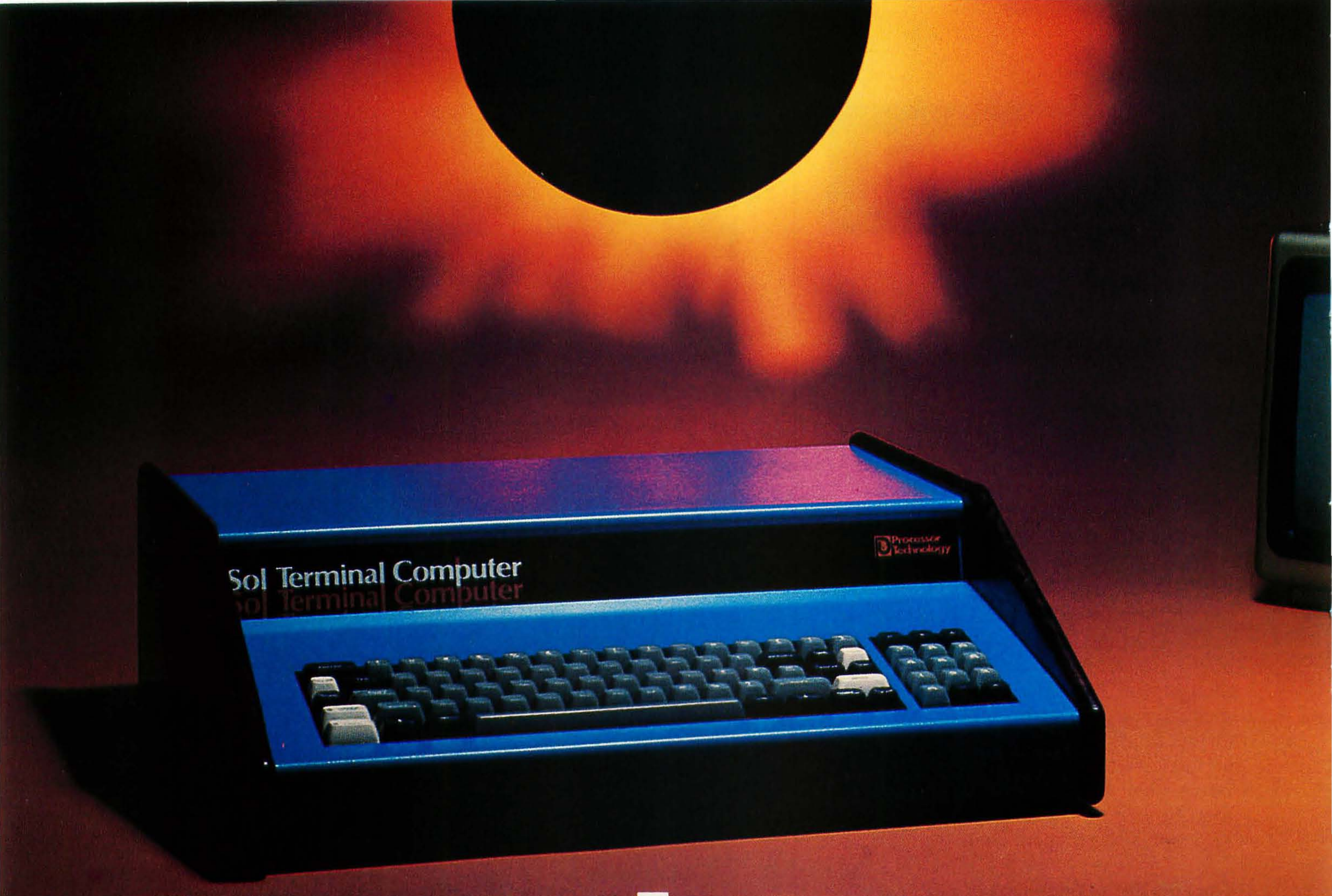
There is a famous rule of thumb which is well-known to philosophers, logicians, mathematicians and natural scientists, which might be called the "Principle of Simplicity." The familiar name for this principle is the rule of "Occam's Razor" after a medieval logician (William of Occam) who has become identified with the concept. Paraphrased from the typical undergraduate

Continued on page 66

Articles Policy

BYTE is continually seeking quality manuscripts written by individuals who are applying personal systems, or who have knowledge which will prove useful to our readers. Manuscripts should have double spaced typewritten texts with wide margins. Numbering sequences should be maintained separately for figures, tables, photos and listings. Figures and tables should be provided on separate sheets of paper. Photos of technical subjects should be taken with uniform lighting, sharp focus and should be supplied in the form of clear glossy black and white or color prints (if you do not have access to quality photography, items to be photographed can be shipped to us in many cases). Computer listings should be supplied using the darkest ribbons possible on new (not recycled) blank white computer forms or bond paper. Where possible, we would like authors to supply a short statement about their background and experience.

Articles which are accepted are typically acknowledged with a binder check 4 to 8 weeks after receipt. Honorariums for articles are based upon the technical quality and suitability for BYTE's readership and are typically \$25 to \$50 per typeset magazine page. We recommend that authors record their name and address information redundantly on materials submitted, and that a return envelope with postage be supplied in the event the article is not accepted.■


Sol Terminal Computer
Processor Technology



capacity 16,384 word memory. It sells for $1883 in kit form; $2283 fully assembled.

For even more demanding tasks, Sol System III features Sol-20/SOLOS, a 32,768 word memory, the video monitor, Helios II Disk Memory System and DISK BASIC Diskette. Price, $4750 in kit form, $5450 fully assembled and tested.

And remember, though we call these small or personal computer systems, they have more power per dollar than anything ever offered. They provide performance comparable with mini-computer systems priced thousands of dollars more.



The functional beauty of Sol Computer Systems is more than skin deep. A look inside reveals a simple elegance of design and sturdy construction.

The Small Computer Catalog for the rest of the real computer system story.

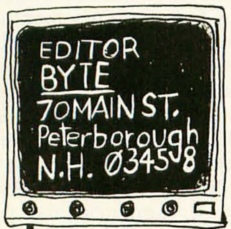
Visit your local computer store for a copy of our fully illustrated 22 page catalog. Or you may write or call us if more convenient. Please address Processor Technology, Box B, 6200 Hollis Street, Emeryville, CA 94608. (415) 652-8080.

Processor Technology Corporation

Circle 104 on inquiry card.

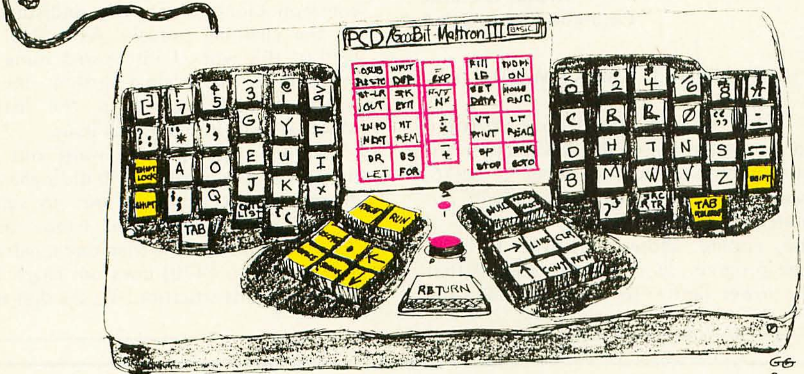
Letters

OPTIMIZING KEYBOARDS?



Dear Mr. Helmers:

In your editoral in the April 1977 BYTE, you made mention of the Dvorak system. By one of those peculiar coincidences, among great minds like ourselves, I was juggling a concept of an ergonomic keyboard - based upon the Dvorak system and the new P.C.D. Ltd.'s [42 Queens Road, Farnborough, Hants, England] Maltron II electric typewriter-Keyboard (see photo, pg. 96 April 1977 Popular Science). Below is a rough version, the gold keys-single function, red-out lined keys-shifted by mode select. Choice of functions arbitrary.



It seems to me before small computer/typewriters become more universal we should investigate the Dvorak + ergonomic designs.

Gary Goodman
P.O. Box 396
Hardinsburg, Ky. 40143
(502) 257-8806

With the ability to redefine keystrokes at will, it is possible to do considerable experimentation with keyboard meaning just within the confines of a standard key layout. The PCD keyboard seen in Popular Science *April 1977 and sketched in your letter represents experimentation with physical placement, a different order of problem.*

HIRED GIRL AND THE MOWBOT

Re: James F Ward (Ask BYTE, BYTE, Volume 2, Number 4, page 59):

There is a robot lawnmower. It is called "Mowbot," and has been written up in magazines such as *Scientific American* and *Esquire*. It is a self-propelled mower that covers the area to be mowed in a random walk. Out-of-bounds areas are marked by a current carrying wire buried just below the surface of the ground; the Mowbot senses the magnetic field around the wire and "bounces" off the boundary in a random direction. If it contacts an object, it stops. Besides not requiring an operator, one claimed advantage is that the random path covered by the machine eliminates undesirable mowing patterns. The time required to cover a given area in a random path is surprisingly not much greater than the time required using a regular path (nonrandom path). This gadget has been marketed for some time. A similar idea could be used for a home vacuum cleaner, although making it small enough to sweep under the furniture would be a tough mechanical problem. Those who were devotees of B grade TV as children will recall a "Science Fiction Theater" story that featured a mysterious couple from the future who used such a gadget in their home. Of course, Dan Davis has already invented it. (See Heinlein, R A, *The Door into Summer*, Signet Books, New York). I bought it when paperbacks were still 50 cents. In the book, "Hired Girl" was first marketed in 1970; however, not in this universe.

William R Hamblen
946 Evans Rd
Nashville TN 37204

MORE INFORMATION ON MOWBOT

Enclosed is a data sheet for the Mowbot automatic lawnmower which I distribute in the Cincinnati OH area. This lawnmower has been around for at least seven years and costs considerably less than the $1500 quoted in the letter from James Ward found on page 59 of April 1977 BYTE.

The lawnmower is completely automatic and will cut around trees, permanent obstacles and patios with no problem. The secret to the Mowbot's performance is in a buried 14 gauge copper wire which is placed around the perimeter of your yard about 1 to 2 inches below the surface. The signal generator which comes with the Mowbot generates a high frequency current which the Mowbot will detect and turn away from. The Mowbot's pattern is completely random which gives your lawn a finished manicured look. And meanwhile, you are off watching TV, playing golf or working on your robot to clean your house (a chore the Mowbot will not presently perform).

I thought you might be interested in this "robot" lawnmower. If I can be of any more assistance, please let me know.

Kim Pollock
11824 Fairsprings Ct
Cincinnati OH 45246

The Mowbot is manufactured by Kistler Products, 9 Hacket Dr, Tonowanda NY 14150. A factory representative in phone conversation said that a limited number are available from stock for $775. The company also markets a swimming pool cleaner which works on a similar principle.

SMALL COMPUTERS

While skimming *EDN* magazine this morning I found an item that brought to mind your April 1977 editorial concerning Arthur Clarke's Minisec. Pages 43 to 46 of the April 20 1977 issue of *EDN* have an article on a microproces-

sor based calibration aid used by the US Navy. Of particular interest is the means of operator interface. Under control of a PACE processor, an 80 by 256 dot matrix plasma display panel displays various control legends and prompting messages as well as graphics, like XY plots. A 16 by 4 infrared sensor-detector array "looks through" the display panel to detect the presence of an operator's finger on any of 64 spots on the panel.

Now, wouldn't it be nifty to use something like that as the front panel of our home computers?

How about the pocket computer? Well, not yet, but today we could build a personal computer about as large as a portable cassette tape recorder. Texas Instruments now supplies samples of a 92 K magnetic bubble memory (complete with all magnetics) in an oversized 14 pin DIP. That's 11.5 K bytes on one chip with nonvolatility, too!

Put together a processor, the panel display/sensor arrray, a bubble memory package or two, and a NICAD battery pack, perhaps even a cassette drive, and you have a truly portable personal computer 300 years ahead of its time.

D Lamkins
D B L Electronics
Canton MA 02021

SURPRISE!

Dear Carl/Karl,

Yes, it's a small universe . . . and I'm embarrassed to say that I'd never even noticed the name of BYTE's editor. . .

Although the January issue is the latest to arrive here by balsa-wood raft, my ever-loving editor at Ballantine Books, Judy-Lynn del Rey, has airmailed me your editorial on *Imperial Earth*, which I read with great interest. I was well aware of the total impossibility of imagining 23rd century technology, and am glad that my "Minisec" isn't already listed in next reason's models. Incidentally, the new Casio CQ-1 *does* contain a clock/calendar of the type I suggested!

However, the COMSOLE is already upon us! Earlier this month I watched the British Post Office engineers demonstrate their VIEWDATA system at the Venice EUROCON 77 (see enclosed leaflet). When fully developed, it will do essentially everything I described in Chapter 24.

Concerning alphanumeric keyboards for a hand-sized device, I agree that you wouldn't use them for extended keystroking. But you'll need them to make brief entries for such purposes as directory and dictionary information (*vide* the problem of the wristwatch telephone *directory*, which I mentioned in my speech at the MIT telephone centennial —you'll find it in the upcoming *The View From Serendip).*

When I was in Boston last year I was fascinated by the video displays in Marvin Minsky's lab, and am determined to take one back to Sri Lanka with me on my next visit in October. . . if I am able to make a choice from your embarrassing selection of riches. (ECD's Micromind seems the one that best fills the bill, as all I *really* want to do is to make pretty pictures and impress the neighbors.)

Arthur C Clarke
Colombo SRI LANKA

BUG REPORT: AMSAT MONITOR

I just wanted to call your attention to a program bug in the AMS80 monitor you published in September of 1976, page 108. The punch routine will not punch out high memory correctly if the ending address is over FFEFH. The program, as it stands, requires that the lower limit +16 be greater than the upper limit at the end of the program to start its last record (03BB DAC303 JC PN1). Changing this instruction to JM PN1 solves this problem.

D K "Billy" Manzarek
8609 Cedros St, #2
Panorama City CA 91402

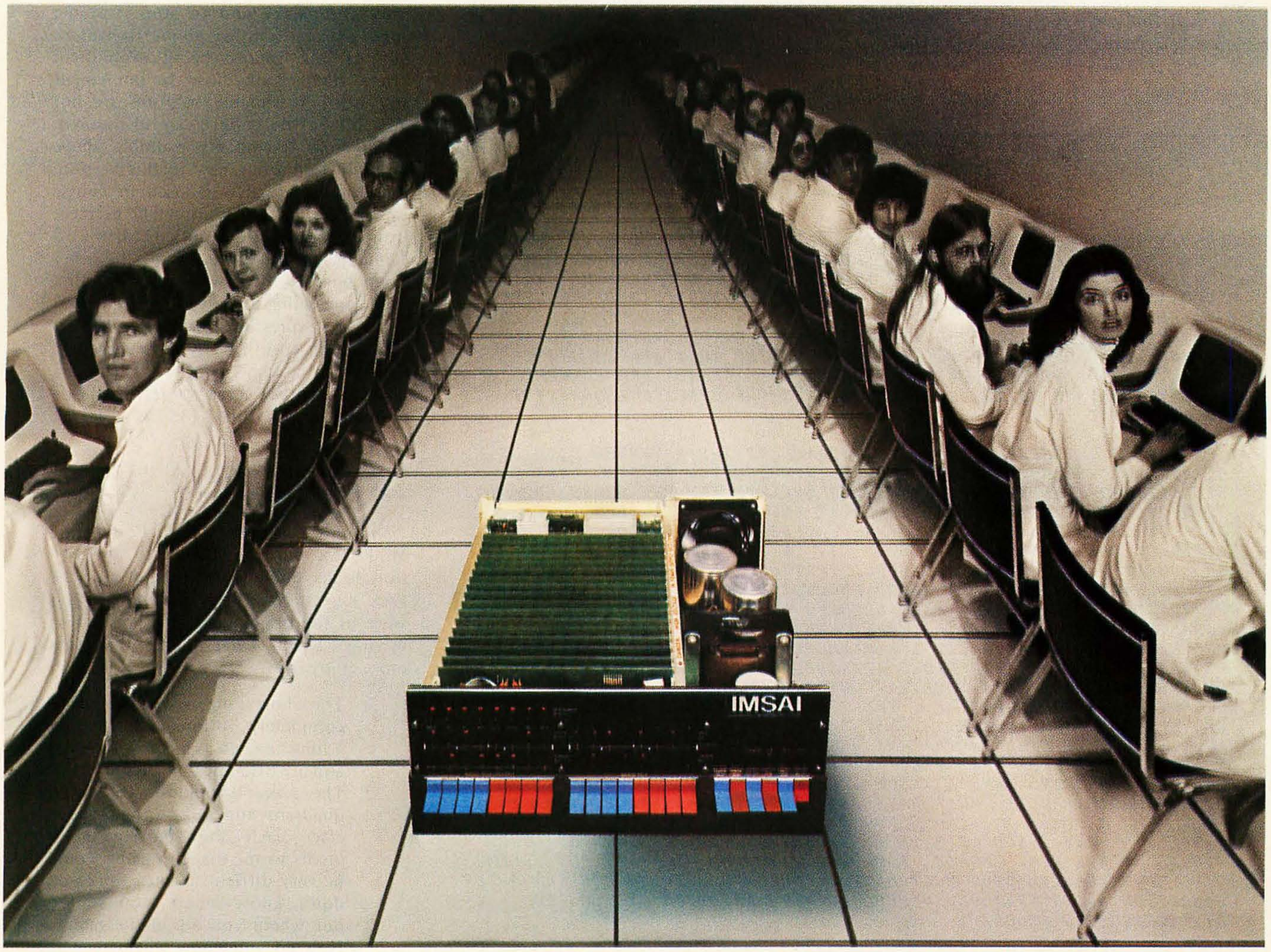
BUG REPORT ON "FLIGHTS OF FANCY . . ."

In the March 1977 issue I found an old friend about whom I'd like to make a few comments. I am referring to "Flights of Fancy with the Enterprise." This version of Star Trek is a modified version (aren't they all) of STTR1, the Star Trek game provided by Hewlett-Packard for their 2000F series timesharing systems. Further, it was a submitted program, so who knows where it came from originally. I ran across it in their program library listings soon after I had discovered computers and programming in high school. Being an optimistic beginner I decided that I would modify it to run on our 2000E series system which did not support formatted output nor have a workspace large enough to hold the entire program.

After some work I succeeded in splitting it into three pieces which could chain (call into execution) to each other, and which communicated through a common block which was undisturbed by the chaining process. As a side effect of this work I discovered some of the bugs in the original version, one of which is still present in the listing which appears in the March issue.

If one attempts to warp out of the galaxy, the program will die a ghastly death with a subscript out of range error. This is because the handy code used if you move more than one quadrant (lines 1360 to 1470) does not check for or correct this situation. To see that this

POWER.



IMSAI Introduces the Megabyte Micro.™

The Megabyte Memory

Until today, the largest memory you could fit and address in a single microcomputer CPU was 65K.

Now, IMSAI presents an incredible memory system for micros 16 times more powerful than yesterday's best.

Imagine, a full megabyte of power from sixteen 65K RAM boards.

And, to control all this, the IMSAI Intelligent Memory Manager (IMM), the super control board.

You can write protect blocks throughout the full megabyte. Or, map in 16K blocks.

Plus, preset 16 mapping configurations with protect for high speed transfer or rapid change.

All interrupts are fully vectored, and there's an interrupt if an attempt is made to write into protected memory.

There's even a real "time of day" clock.

65K, 32K and 16K RAM Boards

Until today, the most memory you could plug into a single slot was 16K.

Now, IMSAI presents memory boards in astonishing multiples of sixteen: 65K, 32K and 16K low power, dynamic RAM Boards. They can be used in any S-100 bus computer individually or in combination to form conventional systems up to 65K bytes.

Every board is fast. With "hidden refresh" and *no* "wait state."

The Complete Megabyte Microcomputer System

The IMSAI Megabyte Micro™ is only part of the story. The full system can include dual floppy disks, terminals, plotters, printers and tape cassettes.

IMSAI also offers the finest high level and peripheral software available. Paper tape and Tape Cassette I/O and super Disk Operating Systems. Plus, BASIC and Disk BASIC with more high level languages coming.

Until today, the microcomputer's potential was just something you talked about.

Now, you can put it to work. Powerfully.

Circle 106 on inquiry card.

GENTLEMEN:
I'm power hungry!
☐ Send 65K RAM Board Kit $2599 ☐ Assembled $3899
☐ Send 32K RAM Board Kit $749 ☐ Assembled $1099
☐ Send 16K RAM Board Kit $449 ☐ Assembled $679
☐ Send IMM ROM Control Kit $299 ☐ Assembled $399
☐ Send IMM EROM Control Kit $499 ☐ Assembled $699
☐ Send full catalog $1.00

Check/MO enclosed. Amt. $ ______
Charge my: ☐ BAC ☐ M/C

______ Exp. Date: ______

Sig. ______

☐ Send name of my nearest IMSAI dealer

Name ______

Company ______ Title ______

Address ______

City ______

State/Zip ______

IMSAI®

IMSAI Manufacturing Corporation
14860 Wicks Blvd.
San Leandro, CA 94577
(415) 483-2093 TWX 910-366-7287

problem really exists, start in quadrant 1, 1 sector 1, 1 and move in direction 3 with a warp factor of 1 (Q1 = 1, Q2 = 1, S1 = 1, S2 = 1, W1 = 1, C1 = 3). You will find that you end up in quadrant −1, 1 sector 8, 1 and the next time an array is indexed with the quadrant coordinates you will have a problem.

Exactly what one should do isn't clear: you can either type a nasty error message and stop the ship at the edge of the galaxy, or you can make the galaxy "circular" so that the ship comes back in on the other side. In my original correction I modified the move section to actually move the ship from one sector to the next, across as many quadrants as necessary to complete the move. This single stepping allowed me to catch the ship in the act of trying to leave the galaxy and to stop it at the edge with a warning about navigation out of the galaxy being prohibited. These changes or those to implement any other reasonable solution are not difficult but do require careful thought.

In hand tracing my example above you will find a bug new to this version: the array C (which is used in course calculations) is not completely defined. In lines 170 and 180, only 11 of the 18 elements are initialized. What the other seven have in them when they are first used (presuming your system will let you evaluate expressions with undefined variables in them) is anyone's guess and will most certainly not be what you want. The other seven values should be defined in say line 185 as follows:

185 C[1, 2] = C[2, 2] = C[6, 1] = C[7, 1] = C[8, 1] = C[8, 2] = C[9, 1] = 1

I don't wish to sound like I'm putting Mr Price's work down; overall, this version looks quite reasonable. I do have two suggestions (both are things that were in the original but are not in this version) which I feel make the game more enjoyable.

1. The short range scans should be completely optional as they take a long time to type on a Teletype and are often not needed.
2. The player should be told what quadrant and sector they are in after each ship movement. It seems to me that this game would be very difficult to play since you don't know where you started nor where you are in the galaxy. How do you know where to look for the rest of the Klingons without wandering around aimlessly hoping to find them? (Not knowing where you've been makes it hard to know where you haven't been.) This lack of information also makes the "flight out of the galaxy" bug worse; if you don't know where you are it's easier to go too far the wrong way.

I also seem to remember that there is a problem in the library computer section, but I don't remember exactly what or where it is. Perhaps it has been corrected in this version. If it comes back to me I'll let you know.

John Stafford
4231 Casa de Machado
La Mesa CA 92041

jws:unix

AN APL BIGOT SPEAKS

DEAR EDITOR;

YOU MAY CONCLUDE FROM THIS LETTER THAT I AM AN APL BIGOT. YOU, OF COURSE, ARE CORRECT. I AM TYPING THIS LETTER ON MY SELECTRIC II USING MY APL ELEMENT (987). I PRESENTLY DO NOT HAVE MY OWN PERSONAL COMPUTER SYSTEM BUT AM LOOKING FORWARD TO THE DAY WHEN I CANBEGIN THAT PROJECT. I USE (OR MISUSE) APL ALMOST DAILY IN MY WORK AND OCCASIONALLY HAVE AN IBM 5100 AT MY DISPOSAL FOR A WEEKEND. I AM OVERJOYED TO SEE THE MOMENTUM THAT APL IS STARTING TO PICK UP HERE IN THE UNITED STATES. CANADA AND EUROPE ARE WAY AHEAD OF US IN ACCEPTING APL AS THE CONCISE PRODUCTIVE LANGUAGE THAT IT IS.

I READ WITH INTEREST THE ARTICLE ON APL IN THE NOVEMBER BYTE BY MR. ARNOLD. THE ARTICLE WAS A GOOD INTRODUCTION TO APL. THE DEFINITION OF THE PRIMITIVES WAS INCOMPLETE AND THE REPRESENTATION OF THE PRIMITIVES WAS CRUDE AT BEST. I WILL TYPE THE COMPLETE CHARACTER SET WITH OVER-STRIKES AT THE BOTTOM OF MY LETTER SO THE READERS CAN SEE IT IN ALL ITS BEAUTY. ON MR. ARNOLDS MICRO-PROCESSOR APL SPECIFICATIONS MY COMMENT IS '..BACK TO THE DRAWING BOARD.' A MICRO-PROCESSOR IS NOT A 360/370 SO WHY ADOPT THEIR FORMAT. ALSO USING 4 BYTES TO REPRESENT A CHARACTER IS RIDICULOUS. I HAVE READ OF AN APL IMPLEMENTATION DONE IN EUROPE ON A 16-BIT 16K MACHINE THAT USED THE COMPLETE APL LANGUAGE. THEY HAD DISKS ATTACHED TO THE MACHINE. I THINK THE ANSWER IS TO USE A COMBINATION OF FLOPPY DISK AND PROM/ROM TO PROVIDE THE INTERPRETER. THE IBM 5100 MANAGES TO PUT THE FULL APL LANGUAGE IN 16K. ISN'T THERE A SIMILAR APPROACH FOR MICRO? (LET ME EXPLAIN THAT MY USE OF PROM/ROM IS NOT BASED ON ANY KNOWLEDGE OF WHAT THEY REALLY ARE OR MEAN. I'VE JUST READ THE TERMS AND HAVE A VAGUE IDEA ABOUT THEM. I AM MORE SOFTWARE ORIENTED. I DON'T HAVE ANY HARDWARE KNOWLEDGE. ..HELP!!)

IN THE FEBRUARY ISSUE OF BYTE I SAW SEVERAL LETTERS ASKING ABOUT FINDING AN APL ROM. AGAIN, I MAY BE WAY OVER MY HEAD, BUT SEEMS TO ME THAT THE IBM 5100 MUST HAVE A ROM OR SOMETHING FOR CHARACTER GENERATION. CAN THAT BE INTERFACED WITH EXISTING SYSTEMS TO GET THE DESIRED RESULT?

FINALLY, THE BYTE STAFF SHOULD BE HUNG FROM THE NEAREST YARDARM. YOU ACCEPTED AN ARTICLE FROM DR. KENNETH B. WELLES WHICH CONTAINED A PICTURE OF A HOME-BREW MODIFIED SELECTRIC TYPEWRITER AND NOT A WORD ON THE CONVERSION. I HAVE READ MANY LETTERS IN BYTE BEGGING FOR INSTRUCTIONS ON HOW TO MAKE THE CONVERSION. IF YOU ACCEPT ANOTHER ARTICLE FROM DR. WELLES WITHOUT A COMPLETE DETAILED DESCRIPTION OF THE SELECTRIC CONVERSION PROCESS, I WILL PERSONALLY COMMIT BATTERY UPON YOUR PERSON.

I ENJOY YOUR PUBLICATION AND AM THE PROUD OWNER OF ALMOST A COMPLETE SET OF BYTE. SOMEHOW I HAVE MISPLACED ISSUE 2 IN THE PROCESS OF MOVING FROM CALIFORNIA TO ILLINOIS. KEEP UP THE GOOD WORK!

SINCERELY,

Gary K. Luther

GARY K. LUTHER
RR 1
ROCHESTER, IL 62563

PS: HERE IS THAT BEAUTIFUL CHARACTER SET I PROMISED.

```
¨¯<≤=≥>≠∨∧-÷ ?ωερ~↑↓ιO*→ α⌈⌊_∇∆∘'⎕() ⊂⊃∩∪⊥⊤|;:\ ⊖⍟⌽⌽⌀⍞⌹⍀≠⍒
1234567890+× QWERTYUIOP← ASDFGHJKL[] ZXCVBNM,./ ⍎⍕⍝!⍒⍋⍱⍲⌶
```

Information on Selectric conversions is coming; have patience. Dan Fylstra's article on the IO Selectrics is a first step, for the electronics of his interface is the same as that required for a converted manual machine. The mechanical details of the interface are discussed in one article presently being edited and a second article promised (but not yet in our office). Both of these future articles are written by people who have successfully accomplished working systems.

HOMEBREW?

First of all, as a newcomer to BYTE, I must say you have a very good set up, but after reading awhile I must call

Continued on page 136



IS YOUR DOLLAR BUYING AS MUCH POWER & FLEXIBILITY AS TDL'S XITAN SYSTEMS PROVIDE?

OUR CUSTOMERS SAY THINGS LIKE THIS ABOUT TDL PRODUCTS:

"... the best CPU board I've put together... excellent parts... worked right off without trouble shooting." .... JRG, Cambridge, Mass.

"Great product." .... KMM, Bella Vista, Ark.

"... high quality components, good engineering & complete documentation... up and running without any problems." .... WP, Seattle, Wash.

"Excellent." .... Col. DWW, Santa Maria, Calif.

"Very impressed with superb quality." .... SK-L, Boston, Mass.

XITAN *alpha 1*

When we combined our highly praised ZPU board and our System Monitor Board, we defined the standard for the industry; we integrated more power and flexibility in two slots of our motherboard than most other systems can muster using five or more boards. When we put this setup into our rugged aluminum case we created the first XITAN system, the *alpha 1*. By adding a CRT terminal and/or teleprinter you will have a complete computer system.

KIT: $769 ASSEMBLED & TESTED: $1039

XITAN *alpha 2*

By adding a Z16 memory module and our PACKAGE A software to the *alpha 1* we created a second XITAN system, the *alpha 2*. Thus, a complete and extremely powerful micro-computer system emerges well worthy of you who are operating at the most sophisticated levels. The XITAN *alpha 2* provides you with 18K of RAM, 2K of ROM, 2 serial I/O ports, 1 parallel I/O port, our 1200 baud audio cassette interface as well as our extraordinarily powerful software package which includes 8K Basic, the Text Output Processor, the Zapple Text Editor and the Relocating Macro-Assembler. Add your own I/O device and GO...with the most powerful and flexible micro-computer package ever offered.

KIT: $1369 ASSEMBLED & TESTED: $1749

IF YOU ARE A BEGINNER, YOU WON'T EASILY OUTGROW THE XITAN SYSTEM.
IF YOU ARE AN ADVANCED USER, YOU WILL DISCOVER XITAN IS EXACTLY WHAT YOU NEED.

Circle 10 on inquiry card.

†Write for descriptive brochure on the XITAN ***alpha*** series and system software. When you ask at your dealer, say "ZY-TAN."

ORDERING INFORMATION: Send check, money order or BankAmericard, Master Charge current number and expiration date. Shipping is usually made via UPS or UPS Blue Label. Specify other arrangements if you wish. Prepaid orders are shipped postpaid.

TECHNICAL DESIGN LABS
Z80
RESEARCH PARK BLDG. H 1101 STATE ROAD
PRINCETON, NEW JERSEY 08540 (609) 921-0321

Circle 107 on inquiry card.


Introducing Apple II.™
DOW JONES INDUSTRIAL AVERAGE
TOTAL DAILY TRADING VOLUME
SHARES

You've just run out of excuses for not owning a personal computer.

Clear the kitchen table. Bring in the color TV. Plug in your new Apple II,* and connect any standard cassette recorder/player. Now you're ready for an evening of discovery in the new world of personal computers.

Only Apple II makes it that easy. It's a



complete, ready to use computer, not a kit. At $1298, it includes video graphics in 15 colors. It includes 8K bytes ROM and 4K bytes RAM—easily expandable to 48K bytes using 16K RAMs (see box). But you don't even need to know a RAM from a ROM to use and enjoy Apple II. For example, it's the first personal computer with a fast version of BASIC permanently stored in ROM. That means you can begin writing your own programs the first evening, even if you've had no previous computer experience.

The familiar typewriter-style keyboard makes it easy to enter your instructions. And your programs can be stored on—and retrieved from—audio cassettes, using the built-in cassette interface, so you can swap with other Apple II users.

You can create dazzling color displays using the unique color graphics commands in Apple BASIC. Write simple programs to display beautiful kaleidoscopic designs. Or invent your own games. Games like PONG—using the game paddles supplied. You can even add the dimension of sound through Apple II's built-in speaker.

But Apple II is more than an advanced, infinitely flexible game machine. Use it to teach your children arithmetic, or spelling for instance. Apple II makes learning fun.

Apple II can also manage household finances, chart the stock market or index recipes, record collections, even control your home environment.

Right now, we're finalizing a peripheral board that will slide into one of the eight available motherboard slots and enable you to compose music electronically. And there will be other peripherals announced soon to allow your Apple II to talk with another Apple II, or to interface to a printer or teletype.



Apple II is designed to grow with you as your skill and experience with computers grows. It is the state of the art in personal computing today, and compatible upgrades and peripherals will keep Apple II in the forefront for years to come.

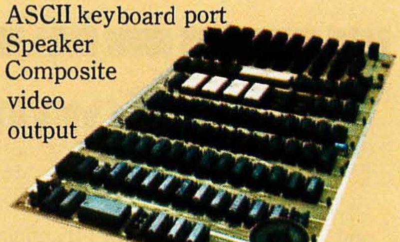
Write us today for our detailed brochure and order form. Or call us for the name and address of the Apple II dealer nearest you. (408) 996-1010. Apple Computer Inc., 20863 Stevens Creek Boulevard, Bldg. B3-C, Cupertino, California 95014.



Apple II™ is a completely self-contained computer system with BASIC in ROM, color graphics, ASCII keyboard, lightweight, efficient switching power supply and molded case. It is supplied with BASIC in ROM, up to 48K bytes of RAM, and with cassette tape, video and game I/O interfaces built-in. Also included are two game paddles and a demonstration cassette.

SPECIFICATIONS

- **Microprocessor:** 6502 (1 MHz).
- **Video Display:** Memory mapped, 5 modes—all Software-selectable:
  - Text—40 characters/line, 24 lines upper case.
  - Color graphics—40h x 48v, 15 colors
  - High-resolution graphics—280h x 192v; black, white, violet, green (12K RAM minimum required)
  - Both graphics modes can be selected to include 4 lines of text at the bottom of the display area.
  - Completely transparent memory access. All color generation done digitally.
- **Memory:** up to 48K bytes on-board RAM (4K supplied)
  - Uses either 4K or new 16K dynamic memory chips
  - Up to 12K ROM (8K supplied)
- **Software**
  - Fast extended integer BASIC in ROM with color graphics commands
  - Extensive monitor in ROM
- **I/O**
  - 1500 bps cassette interface
  - 8-slot motherboard
  - Apple game I/O connector
  - ASCII keyboard port
  - Speaker
  - Composite video output



Apple II is also available in board-only form for the do-it-yourself hobbyist. Has all of the features of the Apple II system, but does not include case, keyboard, power supply or game paddles. $598.

PONG is a trademark of Atari Inc.

*Apple II plugs into any standard TV using an inexpensive modulator (not supplied).

apple computer inc.™

Circle 108 on inquiry card.

Technical Forum

An APL Update

E H Anthony
Zoology Department
University of Guelph
Guelph, Ontario N1G 2W1
CANADA

You and your contributor, Mark Arnold, are to be congratulated for introducing your readers to APL. Nevertheless, the article in your November 1976 issue may benefit from updating with respect to current usage of some fundamental terminology, especially for those who may have encountered APL for the first time in your journal.

Arnold lists functions such as + — X ÷ "Γ"ι but calls them operators. They are functions. Then, when he gets to operators, such as / and . , Arnold calls them composite operators. They are simply operators. Is this not merely nit picking? No, it is not. Is this distinction in terms really important? Yes, it is. To say otherwise is to overlook much that is significant in the evolution of APL.

I believe the corrections proposed here align with the grain of APL in a significant manner. Computer languages scarcely bear close comparison with APL for the simple reason that it is more than just a computer language. First and foremost it is a teaching language. Teaching is the role for which it was born. Requiring a language to be useful in teaching apparently invokes a different and better set of constraints than merely requiring that it enable us to operate a digital computer. In general, it may be said that what is good for teaching is good for all aspects of society. Certainly computer science is not immune to this thesis.

An obvious requirement of a teaching language is consistency in notation and terminology. The lack of such consistency has been a common source of difficulty for those learning traditional mathematics. Common functions like plus and minus have frequently been referred to as operators, while less common ones like sine and cosine have been called functions in the same text. This distinction in terminology tends to divert attention from the basic activity they have in common, namely accepting an (nonfunctional) argument (or arguments) and returning a result. Using terminology that relates them to the same class of activity emphasizes the underlying unity. The term function seems admirably suited to that usage, although the choice is obviously arbitrary.

Operators perform a higher level of manipulation in APL. They are characterized by the fact that they take a function as one or both of their arguments. (Their introduction necessitates a hierarchy in the order of execution, such that operators are executed prior to functions.) The differences between function and operator in APL may be illustrated with reference to the symbol / , which serves dual roles. If we say *1 0 1 1 0/1 2 3 4 5* , we are using the symbol as a compression function. It then has a Boolean vector as left argument and an ordinary numeric vector as a right argument and returns *1 3 4* as a result. Note that neither argument is a function; the only function in the expression is compression, represented by /. If, on the other hand, we say *+/1 2 3 4 5*, we are using / as the reduction operator. It takes as its left argument a primitive scalar dyadic function, ie: a function whose three properties are indigenous to APL (present on the keyboard of an APL terminal), that it is defined initially for scalars (single numbers), and that it accepts two arguments. In this specific instance, the left argument of reduction is the plus function. Plus reduction is summation and the oper-

Continued on page 106



Photo 1: A detail of the actual model, in operation in the authors' Project Solo laboratory at the University of Pittsburgh. The program used to drive the output device is a relatively conventional BASIC version of lunar lander, with modifications to use the analog inputs and outputs diagrammed in figure 1. Input comes from a "throttle" stick which is a lever mounted on a potentiometer read by an analog to digital input conversion channel. Outputs from several digital to analog conversion channels drive display meters (see figure 1), as well as the voltage set point of the motor drive electronics described here. In the authors' system, these analog conversion functions are provided by a Cromemco conversion module which plugs into an Altair (S-100) bus backplane of the computer which runs BASIC for the lunar lander simulation.

Controlling Small DC Motors with Analog Signals

(Or, the Key to Making a Realistic Lunar Lander Simulation Display)

Leon Sweer
Thomas Dwyer
Margot Critchfield
Project Solo
University of Pittsburgh
Pittsburgh PA 15260

Microcomputers are starting to show up in an unexpected variety of places: engineering labs, businesses, factories, golf courses, radio stations, the home, and most recently, the classrooms of adventuresome teachers.

One consequence of this growth has been a rapid expansion in what might be called "blue sky" thinking about how computers might be used. Amateurs aren't afraid to fantasize a bit, knowing that's where half the fun lies. Even when these ideas are too ambitious for immediate implementation, they often suggest simpler partial projects that will eventually fit in as components of a bigger system. For example, getting simple black and white graphics to work on CRT monitors could be the prelude to running a more ambitious sound and color "spectacular" under computer control.

Our experience at Soloworks in applying computers to education indicates that working at this imaginative level helps foster the natural ability of people to learn through creative explorations. This in turn suggests that the more flexible the technology, the better the learning. Computers hold special promise because of the way they can be "transformed" with software. Their main limitation has been at the interface level. For example, it has not been easy to find hardware for controlling "things" (as opposed to symbols on a terminal).

An important first step in solving this problem has been the appearance of analog to digital and digital to analog interface boards (eg: the Cromemco D+7A). On the input side, continuous joystick control now becomes easy. But what about output? In particular, it now makes sense to ask if a computer can be used for control of continuous DC motors. We'll also want to ask if computers can move things according to algorithms that specify position. The answer is yes, provided that (a) we can find a way to amplify the analog output so it can drive a motor, and (b) we can create a feedback path that lets the motor know if it has run long (or fast) enough to accomplish the task at hand.

This article describes one simple approach to solving these two problems. In order to make the ideas involved easier to follow, they are presented in terms of a specific project, namely building a one-dimensional dynamic plotter for use with a "lunar lander" style game. However, with a little imagination, many other applications of these ideas should be possible, including the control of several motors.

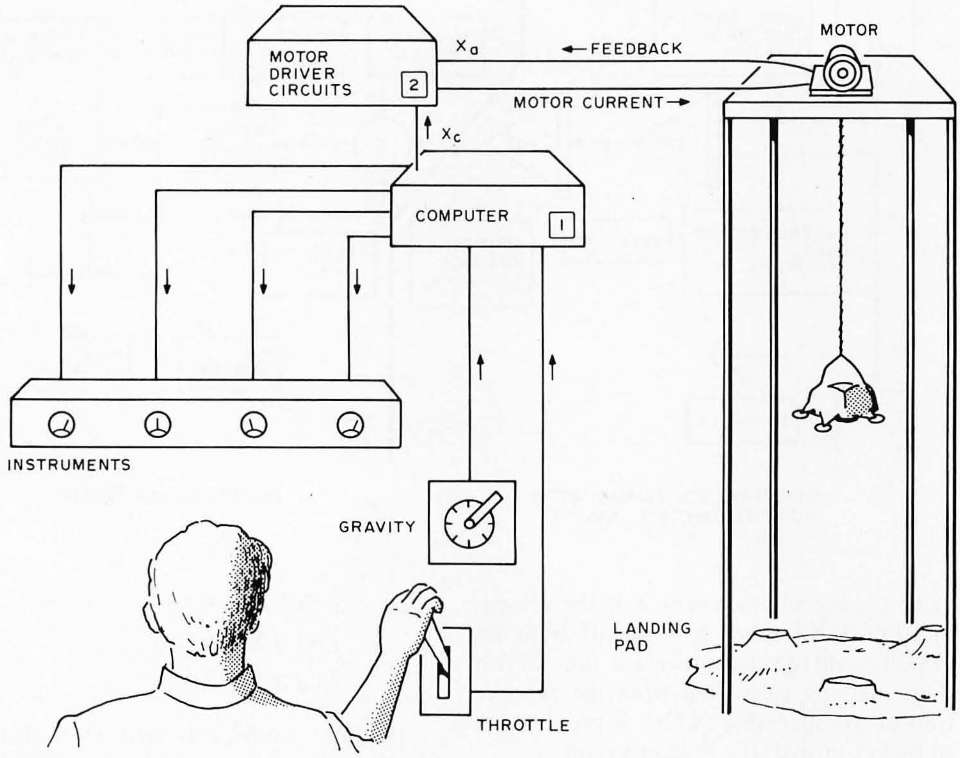


Figure 1: A pictorial diagram of how the one-dimensional plotter fits into the implementation of a planetary landing simulator. The motor drive circuits (2) use feedback to set the altitude of the model ship based on the current set point command X_C issued by a digital to analog conversion output of the computer. The computer (1) implements a mathematical model of the spaceship's flight based on analog to digital conversion of a throttle signal input, with analog output of the ship's altitude as voltage X_C.

The One-Dimensional Plotter

A plotter is an electro-mechanical instrument for displaying the position or motion of some object, usually as a function of time. It can be used for such things as showing the flight path of an airplane or the trajectory of a rocket. To display the position of an airplane it would be necessary to have a three-dimensional display or a three-dimensional plotter (x, y and z, where z is altitude). To show the flight path of a plane flying at constant altitude only a two-dimensional plotter would be needed. The graph produced would have one axis corresponding to the north-south component of motion, while the other axis would represent the east-west component.

While most general plotting work is done in two dimensions with the mechanism dragging a pen to create a picture on paper, a one-dimensional plotter also has applications. For example, a one-dimensional plotter could be used to show the vertical descent phase of a spaceship making a simulated landing on a moon or planet. A small scale model of the spaceship could be raised and lowered a number of feet with respect to a landing surface (the planet's surface) by a motor-driven derrick-like arrangement. The result would be a dynamic plotter that allowed the "pilot" to practice landings in one-dimension, up-down (see figure 1). Later, other components of motion (dimensions) could be added by using a fancier mechanical setup.

Authors' note: *The next few paragraphs can be read over rapidly by those who are not familiar with calculus.*

In this design, the information displayed, either graphically or dynamically by a plotter, is the position of the spaceship or other object of interest. The plotter must be told the initial position, and any changes in position as part of the mathematical model of a spaceship landing. Sometimes this change information is not known directly, and must be calculated from other information, such as velocity. For the spaceship problem, the change in position (called displacement or ΔX) can be calculated from velocity by integrating the velocity with respect to time $\Delta X = \int_0^T v\,dt$. For a constant

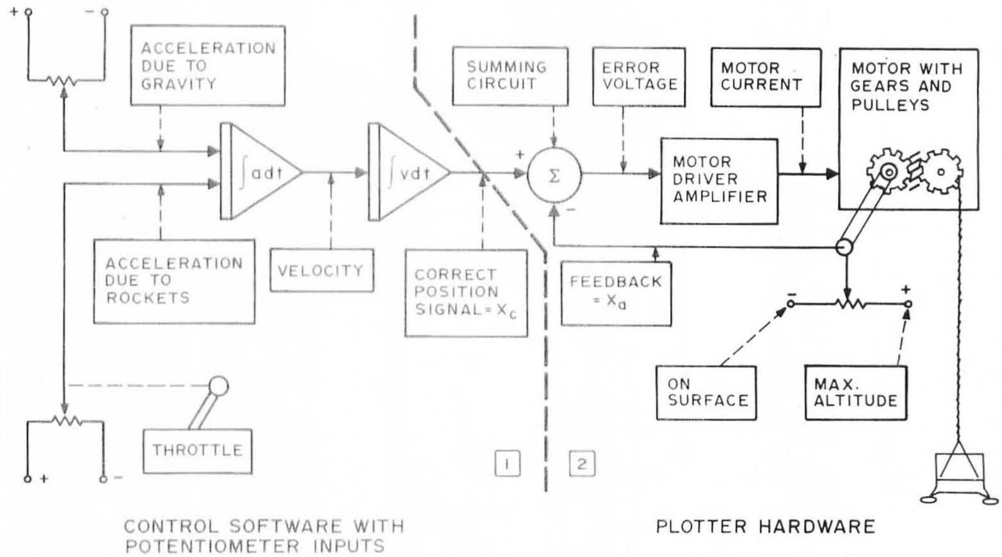


Figure 2: A block diagram of the ship's position display system (2) and the programming model of the ship's behavior which drives the display plotting device (1).

velocity, the displacement X is the velocity multiplied by time, X = vT. If both the velocity and displacement are not known, they may be calculated from the acceleration of the spaceship (which is proportional to rocket thrust). If a = acceleration,

$$V = \int_0^T a\,dt \text{ (assuming velocity starts out as zero)}$$

and therefore,

$$\Delta x = \int_0^T \left(\int_0^T a\,dt\right) dt.$$

These mathematical integrations can be performed with analog circuits, or with a digital computer using numerical integration methods.

Once the displacement of the ship is derived from the acceleration or velocity (using either kind of computer), it can be used to supply the input to the one-dimensional altitude plotter. The job of the plotter is to move the model ship to a position equal to its initial position plus the displacement. Let's call this correct position X_c, and think of it as an input signal to the plotter. X_c, the correct position input, is where the ship should be, according to the equations of motion, in one-dimensional space. If a digital computer is being used, then X_c will have to be generated with one digital to analog output port, using the current state of its programmed model of the ship's behavior. Figures 1 and 2 show how these ideas all fit together.

If we call the actual position of the model ship X_a, this may differ at times from the calculated position X_c. This error is represented by an error signal which expresses the difference between the calculated position of the ship (X_c) and the actual position of the model (X_a).

$\epsilon=0$ if $X_c - X_a = 0$

$\epsilon>0$ if $X_c - X_a > 0$

$\epsilon<0$ if $X_c - X_a < 0$

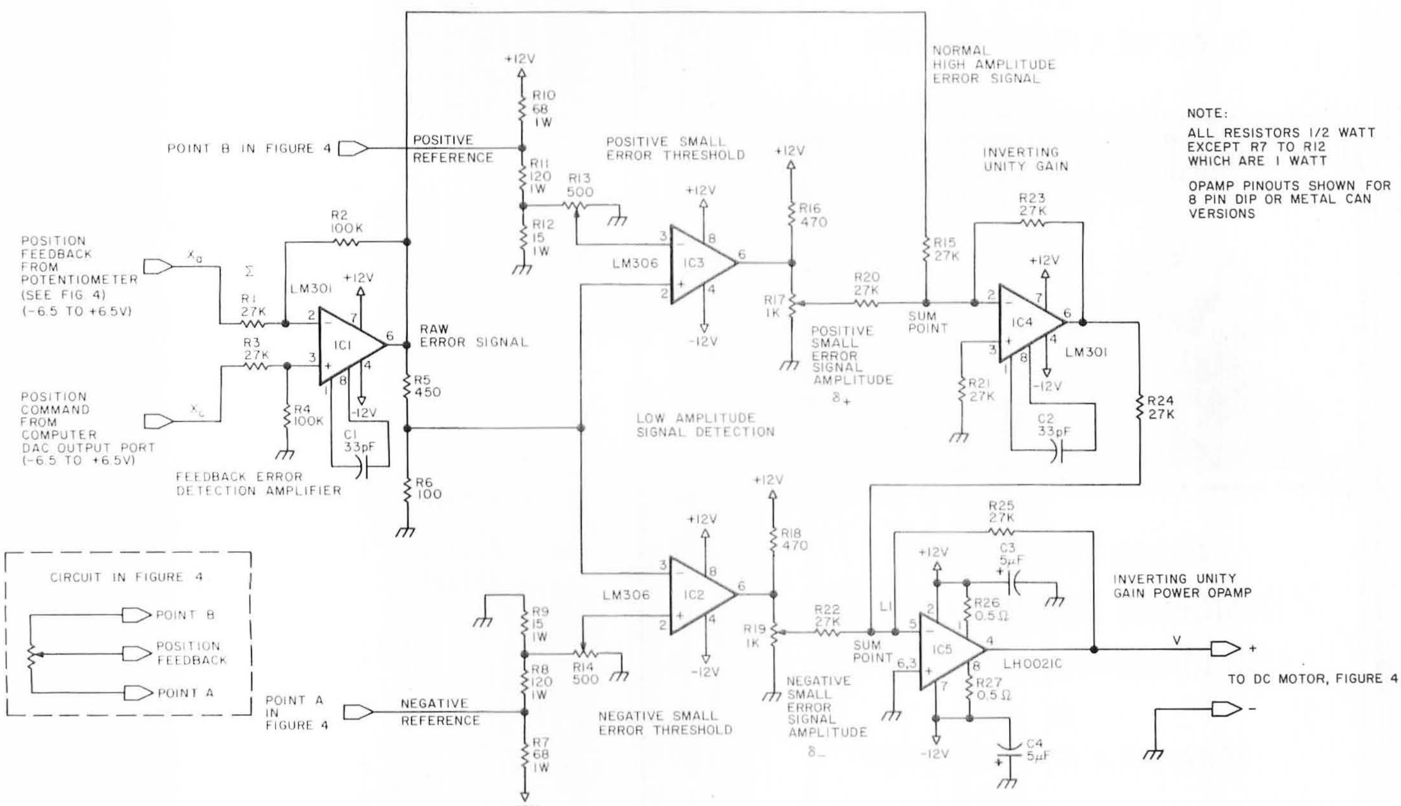
In other words, ϵ is zero when there is no error ($X_c = X_a$), positive when $X_c > X_a$, and negative when $X_c < X_a$.

The error signal ϵ tells the motor which direction to turn in order to move the model ship to the correct position. If ϵ is positive the motor in our diagram of figure 1 must turn one way to raise the model until the actual position becomes equal to the correct position. If ϵ is negative the motor must turn the opposite way to lower the model to the correct position. If $\epsilon=0$ then the correct position equals the actual position of the model and the motor does not turn. The magnitude of ϵ can also be used to determine the speed of the motor.

How does the motor driver know the actual position (X_a) of the model? When the motor moves the model, the motor also moves a wiper on a "feedback" potentiometer. The voltage at this wiper will be proportional to the altitude of the model and may range anywhere from −6.5 V to +6.5 V in this design. The potentiometer feeds back an electronic signal to the "summing" circuit (circuit Σ in figure 3, IC1) indicating the actual position of the model. If the ship is at maximum altitude, the wiper is near the top of the potentiometer, and the voltage is about 6.5 V. If the ship is at minimum altitude, the wiper is near the bottom of the potentiometer and the voltage is about −6.5 V. The mechanical arrangement (see figure 4) must be set up to ensure this relationship.

In summary, the correct position (X_c) is compared by the Σ circuit which tells the motor driver if X_c is greater than (>), less

Figure 3: Details of the motor drive servo circuit. This basic circuit can be used (with appropriate choices of gearing and mechanical arrangement at the feedback pots) to control a large number of mechanisms which depend upon DC motors to set position. The feedback from the position measuring potentiometer X_a is compared with the set point X_c by amplifier IC1. The error signal resulting (if large enough) directly drives the motor after inversion by IC4 when the set point X_c is far away from the actual position X_a. As the set point is approached, the direct error signal becomes nearly zero, so the amplified error signals for small positive or negative displacements (IC2 and IC3) become the dominant terms in the summation at the input of IC5, until finally X_a matches X_c within limits set by the small error threshold potentiometers. This complexity of the circuit is required by the fact that the DC motors require a certain minimum drive voltage in order for any motion to occur.



than (<), or equal to (=) X_a. The motor driver, in turn, tells the motor to go one way, or the opposite way, or not to turn, by a +, –, or 0 voltage. The motor then moves the model ship towards a more correct position, if necessary, while informing the Σ circuit of the new position. When the actual position equals the correct position, the error signal becomes equal to 0 and the motor stops. Any further change in the input signal (correct position signal, X_c) will cause further movement of the model in the same way.

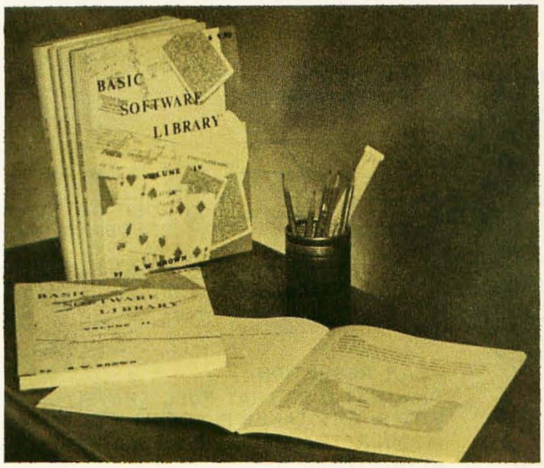
The remainder of this article describes how to build the sector in figure 1 labeled motor drive circuits [2], and how these circuits interface to the other components. This corresponds to the elements to the right of the dotted line in figure 2.

Circuit Description

Figure 3 shows the motor driver circuit, and figure 4 shows the feedback potentiometer and how it is connected to the motor. The output of the potentiometer is the position feedback. The job of the circuit of figure 3 is to attempt to make the position feedback voltage equal to the position command voltage by moving the motor until the difference between the two is zero. This is accomplished as follows.

IC1 is connected as a difference amplifier so that its output is equal to approximately $4*\epsilon$, where ϵ = (position command X_c) – (position feedback X_a). This difference is fed through R15 to the summing amplifier IC4 so that its magnitude is one component of the final motor voltage. Thus, the larger the difference, the larger the voltage, and the faster the motor turns. In actual practice, as this difference approaches 0, it would not be enough to drive a 12 VDC motor. Here is where comparators IC2 and IC3 come in. They are constantly looking to see if the difference voltage, a portion of which appears across R6, is larger than some small positive voltage (set by R13), or smaller than some small negative voltage (set by R14). If

Application Software!



You can buy software from anybody but ours works in your system.

We only sell one product, Quality. Our general ledger does what everybody else's Payroll, A/R, A/P, Inventory, General Ledger, and Tax programs do and it's much shorter and easier to use.

We have been in business for over nine years building a reputation for providing a quality product at nominal prices — NOT what the traffic will bear. Our software is:

- **Versatile** — as most programs allow for multiple modes of operation.
- **Tutorial** — as each program is self prompting and leads you through the program (most have very detailed instructions contained right in their source code).
- **Comprehensive** — as an example our PSD program not only computes Power Spectral Densities but also includes FFT's, Inverse-transforms, Windowing, Sliding Windows, simultaneous FFT's variable data sizes, etc. and as a last word our software is:
- **Readable** — as all of our programs are reproduced full size for ease in reading.
- **Virtually Machine Independent** — these programs are written in a subset of Dartmouth Basic but are not oriented for any one particular system. Just in case your Basic might not use one of our functions we have included an appendix in Volume V which gives conversion algorithms for 19 different Basic's; thats right, just look it up and make the substitution for your particular version. If you would like to convert your favorite program in to Fortran or APL or any other language, the appendix in Volume II will define the statements and their parameters as used in our programs.

Unfortunately this much versatility, comprehensiveness and compatibility requires extra money. But over 85% of our programs in the first five volumes will store and execute, as is, in most 8K Basic's with 16K of free user RAM. If you only have 4K Basic, because of its lack of string functions only about 60% of our programs in Volumes I thru V would be useable, however they should execute in only 8K of user RAM. Volume VI will require the user to have external data files and 30K of user RAM.

Our "General Ledger Program" (Vol. VI) is a complete comprehensive business system designed to keep ALL of your company's records without the need for updating from other programs and there is NO need to keep monthly tear sheets to be added together for end of the year reports as our system will provide you with year end account totals. This program generates over 30 major reports, including: 941's, P/L's, Balance Sheets, and year end account totals for filing Federal Income Tax Schedule C's and/or 1120's.

All of our programs are available on machine readable media. For those that have specific needs, we can tailor any of our programs for you or we can write one to fit your specific needs.

Future additions:

Soon to be released! A "fantastic" word processing package set up for lawyers, publishers and writers; and a Medical Billing system which will also allow a patients record to be individually scanned. AND there are rumors that a 12K chess game, in Basic, will also be released.

Subjects Covered in Volumes I - VI

Vol. I — $24.95 Bookkeeping Games Pictures	**Vol. III — $39.95** Advanced Business Billing, Inventory Investments Payroll	**Vol. VI — $49.95** Complete Business Systems General Ledger Taxes (requires ext. files)
Vol. II — $24.95 Math/Engineering Plotting/Statistics Basic Statement Def.	**Vol. IV — $9.95** General Purpose **Vol. V — $9.95** Experimenter's Program	

AVAILABLE AT MOST COMPUTER STORES

Add $1.50 per volume handling, all domestic shipments sent U.P.S. except APO and P.O. Box which go parcel post. Foreign orders add $8.00/volume for air shipment and make payable in U.S. dollars only.

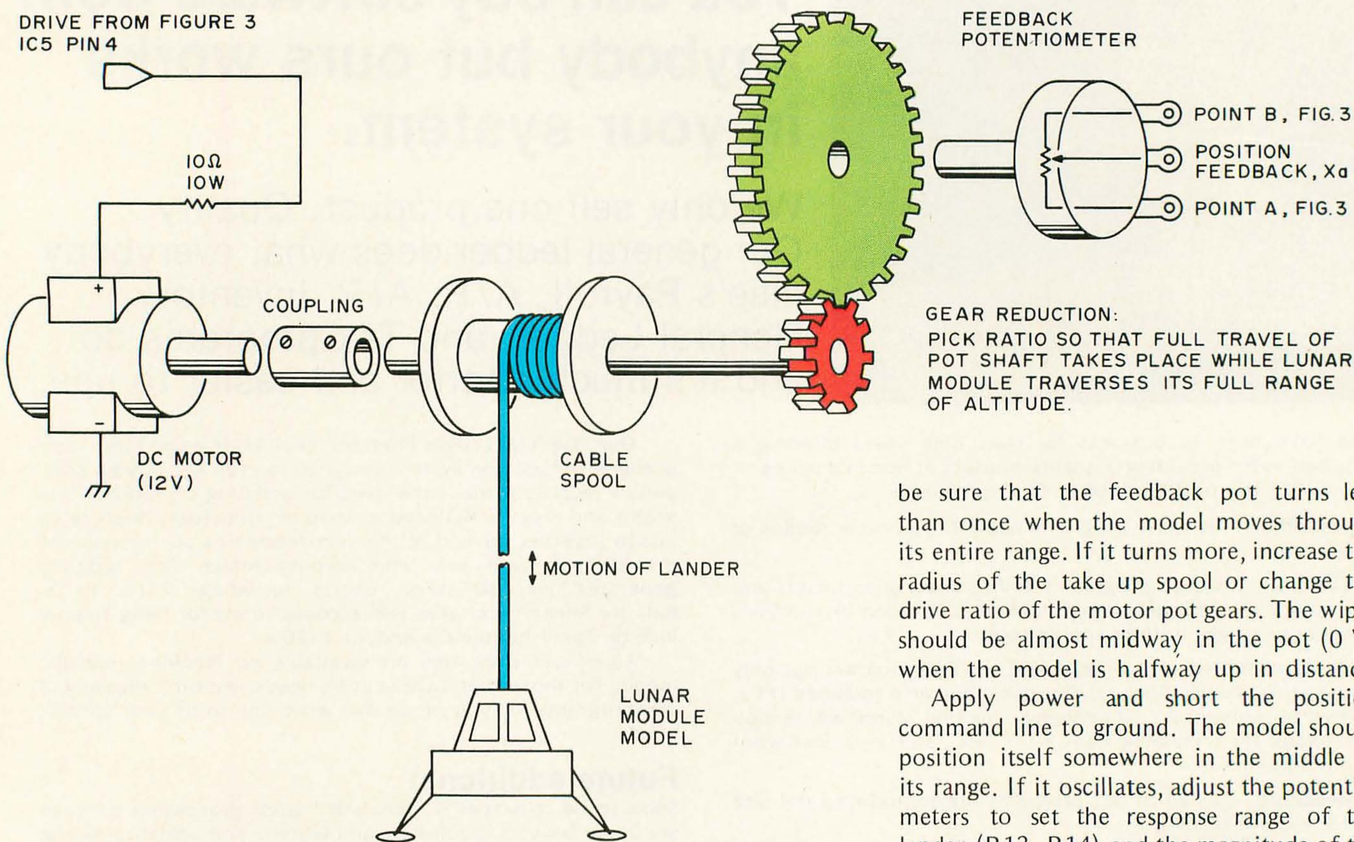
SCIENTIFIC RESEARCH

1712-B Farmington Court
Crofton, Maryland 21114

Phone orders call (800) 638-9194
Information and Maryland residents call (800) 721-1148

master charge — BANKAMERICARD accepted

Figure 4: A pictorial schematic of the mechanical arrangement used for the lunar lander model. A main shaft is connected directly to the motor and has a spool of thread used to suspend the model. The direction of the motor controls whether the module rises or falls relative to the ground surface of the planet model as the shaft rotates. The main shaft has a spur gear which drives a large gear to achieve a gear reduction which maps complete physical travel of the model lander into the motion of the pot's wiper through its complete range.



one of these conditions is satisfied, the comparator output becomes 12 V, and a portion of that output (usually around 2 V, and determined by R17 and R19) drives the motor into its operating range. The actual values of these potentiometers will depend upon the particular motor you use, and upon your actual physical configuration.

IC5 is a high power operational amplifier capable of delivering up to one amp of current at up to 12 V. It is also used as a summing amplifier to bring together the voltage commands from the other circuits.

Parts Substitution and Adjustment

None of the parts are extremely critical, and substitutions can be made. Almost any operational amplifier that accepts +12 V can be used (eg: 741C), and it is not necessary to use the LM306 comparators. LM311s, for instance, will do. A substitute for the power operational amplifier can also be used, but be sure that whatever you use is provided with a good heat sink.

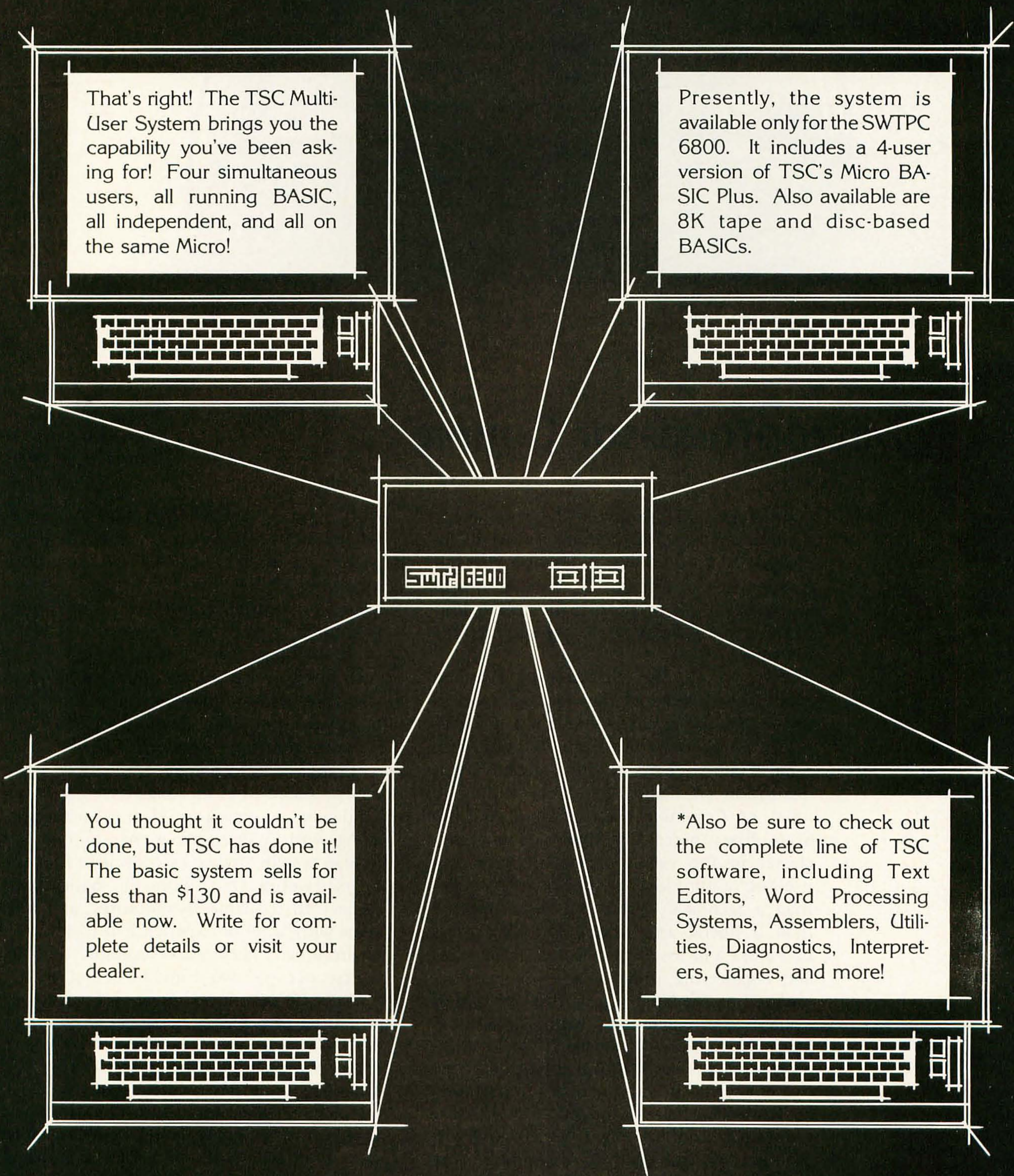
When the entire configuration is set up, be sure that the feedback pot turns less than once when the model moves through its entire range. If it turns more, increase the radius of the take up spool or change the drive ratio of the motor pot gears. The wiper should be almost midway in the pot (0 V) when the model is halfway up in distance.

Apply power and short the position command line to ground. The model should position itself somewhere in the middle of its range. If it oscillates, adjust the potentiometers to set the response range of the lander (R13, R14) and the magnitude of the threshold motor voltage (R17, R19). Experimenting with different motors and gearing setups should give you an idea of what type of performance to expect. Basically, you want the motor to move at a rapid rate if the difference between the X_c and X_a is more than about 2 V, and then to slow down as $X_c - X_a$ approaches zero.

Use a power supply to apply different position command voltages to see if the model "seeks" correctly. Don't apply more than about 6.5 V. If you do, the system will try to move the model so that the position feedback is more than 6.5 V, and this is impossible. Result: the model goes crazy.

Once all of the above is working, you're ready to connect it to an integrating computer. The two choices here are a special purpose analog computer, or a general purpose digital computer with a digital to analog output board. A discussion of what is involved in using a digital computer for this purpose can be found in chapter 10 of the book *An Amateur's Guide to Personal Computing* (by Margot Critchfield and Thomas Dwyer, Addison-Wesley Company, Reading MA, 1977).■

4 USERS ON 1 MICRO?!



Technical Systems Consultants, Inc., P.O. Box 2574, West Lafayette IN 47906

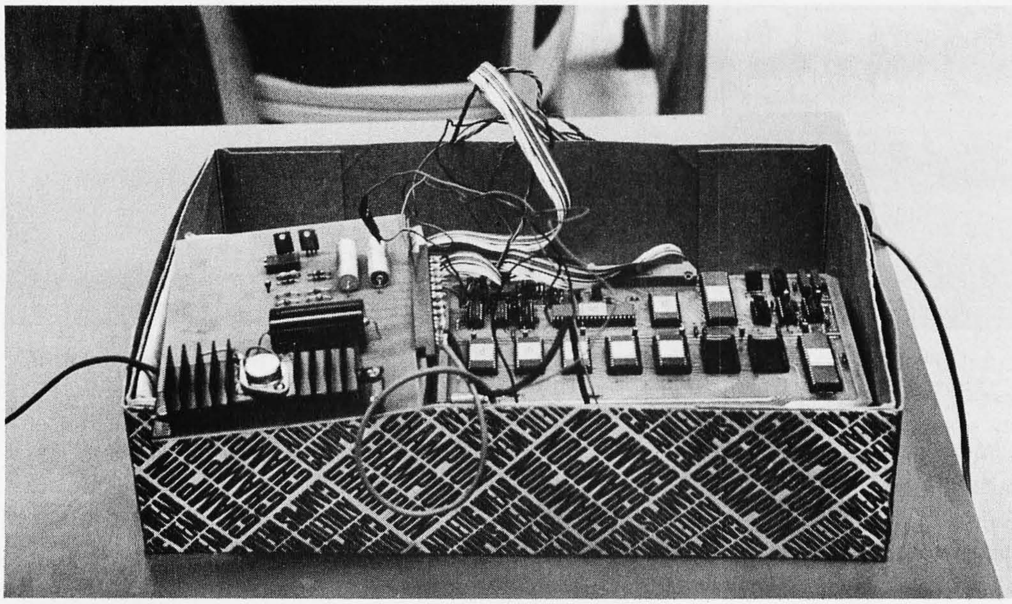


Photo 1: Packaging technology at Franklin. Although it looks a little shaky it still works.

A Microprocessor Course

Mark E Fohl
3567 Karikal Dr
Westerville OH 43081

In August 1975 Franklin University made a commitment to offer a program in microprocessors. I would like to describe what has happened since then, both in terms of Franklin's goals in offering the course and of my experience in developing and teaching the course.

First of all, the character of Franklin may explain some of the rationale used in organizing the course. Franklin University is located in downtown Columbus OH. With some 4,000 students, it is the second largest institution of higher education within 50 miles of Columbus, second only to Ohio State. Sixty percent of the students attend classes in the evening. Franklin promotes itself as "a third kind of education": broader than a technical school but more detailed than a liberal arts school. Franklin offers bachelors and associates degrees in business and engineering, and has recently been accredited by the North Central Association of Colleges and Secondary Schools. Franklin keeps in touch with the needs of local industry and stresses practical education.

I am currently a software design engineer at North Electric Company, located near Columbus. Prior to this, I worked at a government division of RCA doing mostly design automation work. One of my projects there was coordinating the design automation efforts for the design and production of a 3 chip CMOS microprocessor. While at North, I taught math part-time at a small school about 40 miles north of Columbus. I wanted to teach closer to home, so I blanketed the schools in Columbus with a letter asking for a teaching position in math, programming or electronics. Dean Sidney Kelly of Franklin responded that there were no such positions open there, but that Franklin was considering offering a course in microprocessors. Following an afternoon interview, we both thought it was worth a try.

Franklin decided that its microprocessor course would offer thorough coverage of the fundamental concepts of microprocessors along with in-depth experience on one particular system. It was also decided that the course would start with the basics with respect to available prerequisites, and continue with hands on experience in a laboratory environment. These goals suggested that the course occupy two trimesters, and that it actually be a 2 course sequence. Each course would have four class hours per week for 15 weeks, and each course would be worth three semester credits. Furthermore, the sequence would be divided equally between lecture and laboratory, with almost all lecture in the beginning, gradually changing to almost all lab in the end. The course prerequisites were listed as a familiarity with digital logic and the binary number system. These prerequisites could be met with Franklin's series of electronics courses, and



It's like adding a room to your brain!

The System 8813 from PolyMorphic Systems is a complete, powerful problem solver in a single walnut cabinet. This machine allows you to perform complex financial, engineering, and scientific models in the comfort of your office or den.

The high speed video display presents your results in text, tables and graphics. The detachable typewriter-like keyboard permits relaxed program entry and operation. Convenient mini-discs store programs and data for compact filing, secure storage, and fast access. Our disc BASIC programming language is simple enough for the computer newcomer, yet powerful enough to amaze the most advanced users.

The whole family can immediately use and enjoy pre-programmed applications and educational packages. Let the System 8813 become your trusted business, profession, and personal tool. The PolyMorphic Disc System is a completely assembled and tested unit with brushed aluminum front panel, walnut cover, detachable keyboard, video monitor, 16K RAM. Includes system software and fully extended BASIC on disc.

System with 1 disc drive	— $3250
System with 2 disc drives	— 3840
System with 3 disc drives	— 4430

Delivery 60 days ARO. Upgrade packages for POLY 88 owners will also be available. Prices and specifications subject to change without notice.

PolyMorphic Systems (805) 967-0468
460 Ward Drive, Santa Barbara, CA 93111

with a course entitled "Computer Calculations." The sequence would be offered not only to full-time students at Franklin, but also to anyone in the community who, for either industrial or personal reasons, wanted a solid background in microprocessors. Students would need no previous experience with microprocessors, and the sequence would be offered as an attractive alternative to a 3 day seminar.

With the philosophies and preliminaries taken care of, implementation of the sequence remained. By this time it was late August, and the sequence was scheduled to begin the following January. Being the instructor, I was responsible for putting the entire package together. Two tasks presented themselves: the first was to adopt a course outline and prepare corresponding lectures, and the second was to equip the laboratory.

The first task was relatively easy. The obvious strategy was to start from the absolute basics and work upward. Accordingly, the course would begin with a review of binary numbers and binary operations. A review of digital logic elements, both gates and flip flops, would be next, followed by development of a generalized arithmetic and logic unit. The arithmetic and logic unit would proceed to a generalized microprocessor unit, followed by a discussion of a particular microprocessor. Course development could not proceed until specific laboratory equipment was selected.

One of the problems at this point was selection of a suitable textbook. No general textbooks on microprocessors were available at the time. I chose Blakeslee's *Digital Design with Standard MSI and LSI* because a few of its chapters covered microprocessors and microcomputer system elements. In retrospect, the text was not very good for the sequence because of its constant emphasis on hardware design considerations.

While preparing the lectures, I began the process of acquiring laboratory equipment. At the time, the two manufacturers who seemed to be far ahead of the pack were Intel and Motorola. I thought that these two would be in the best position to donate equipment, considering their stable positions in the marketplace and their competitiveness. Accordingly, I contacted local representatives from both Intel and Motorola. I asked each for any help they could provide, and I reminded each that I was talking to the other.

The Intel representative stated flatly that there was nothing he could do personally, but he advised me of a university donation program. I wrote to Intel in California requesting more information and an application form. The material came shortly thereafter, and it turned out that each application was good for up to four kits, each containing an 8080 chip, 256 words of programmable memory, 1 K of blank erasable read only memory, and four 8212 data latch chips. I made a copy of the application and gave it to another instructor at Franklin. We sent in our applications sometime in September; in mid-November, we received eight kits in the mail. Each kit contained the chips plus one programmer reference card.

The Motorola representative said he didn't know exactly what to do, but he was sure there was something. After much correspondence (among him, me, Austin and Phoenix), he said Motorola would provide five Microcomputer Evaluation Kits, without read only memories, if the University would purchase four kits. Each of these kits contained an M6800, 128 words of programmable memory, two PIAs, one ACIA, a printed circuit board, 1 K of read only memory containing the MIKBUG debugging program, assembly instructions, a reference manual, and an applications manual. This offer came in early December.

It appeared that the cash values of Intel's and Motorola's donations were about equal. Intel's equipment was already in, and I had to decide how to proceed. Whereas Intel's donation represented more raw computing power, Motorola's contribution was a better package, which included a tested system design, software and a Teletype interface. Considering that I would have to do the system design for the Intel package, and considering that I was up to my ears already, I opted for the Motorola deal.

The time came now to spend some money. Class was scheduled to start in about four weeks, but we wouldn't be using the lab equipment until about the seventh or eighth week. The dean was reluctant at first to spend any money, but at this point there was no alternative. Besides, the tuition from the students already enrolled in the course was enough to cover both my salary and the lab equipment. The dean gave his blessing, and we proceeded from there. I contacted the Motorola representative and informed him of our decision; then I called the local distributor and ordered the equipment. I expected the equipment to be ready in about two to three weeks, about the same time as the beginning of the winter trimester.

The equipment I ordered from the distributor included all the discrete circuitry specified in the assembly instructions. There were still two glaring omissions from the microcomputer system: the power supply and the IO device. In addition, there was the

Continued on page 90

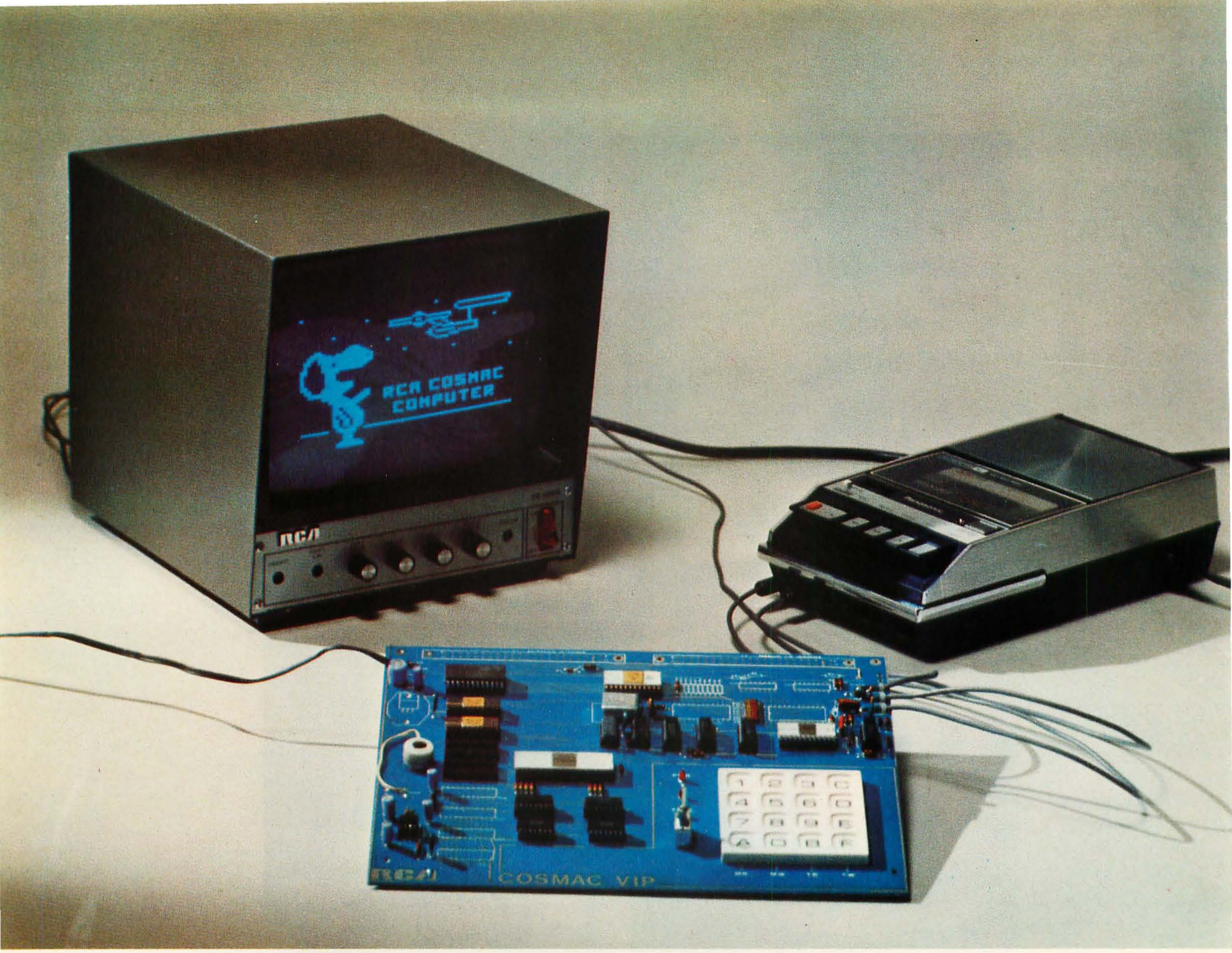


Photo 1: The COSMAC VIP system demonstrating its graphics abilities. The kit version of this product includes the board alone; the user must supply a standard video monitor (or modified TV) and inexpensive tape recorder.

COSMAC VIP, the RCA Fun Machine

Joseph A Weisbecker
1220 Wayne Av
Erlton, Cherry Hill NJ 08002

Most beginners think of computers only in terms of solving equations or processing data in English language format. Ask someone with limited exposure to computers about their use in homes and they invariably mention kitchen recipe files, meal planning, income tax preparation or checkbook balancing. I don't think any of these applications has really caught on so far. This view of computers as super calculators or data processors is apparently shared by many people currently involved in the home computer movement. This orientation keeps prices higher than necessary for the majority of potential hobby users who could have just as much fun with much less sophisticated machines.

The COSMAC VIP was created to provide a viable alternative to expensive hobby computers. It was not designed to compete

with existing systems, but rather to maximize the number of people who can afford to own their own high technology graphics and games fun machine.

In this article I'll give you some background information, outline my design philosophy, and describe the COSMAC VIP system. You can then decide whether this article represents a blatant sales pitch or an honest explanation of a relatively low cost approach to small personal computers. Home or business users who want extensive data processing capability should stop reading now. You will be better off with one of the more expensive BASIC language oriented systems. Those who are beginners or are interested in computers for fun at absolute minimum cost should continue reading.

First some background. About five years ago I developed a system called FRED. This was the prototype for a low cost microcomputer aimed at video games, hobby and school use. (FRED was described in the August 1974 issue of the *IEEE Computer* magazine.) COSMAC VIP evolved from this FRED system. My family and friends have been using this type of system for almost five years and we've only scratched the surface of applications for it. In fact, my $2000 personal BASIC system seems dull by comparison and is gathering dust. The development of the 1 card COSMAC VIP system is due primarily to a desire to share fun I've been having with as many others as possible. RCA deserves a lot of credit for supporting the development and making the system available for home and school use.

Design Philosophy

COSMAC VIP is based on the premise that a home computer should be low cost, easy to use and fun for the whole family. It is therefore aimed at video graphics and games applications. Even the youngest family member can have hours of fun drawing pictures on the TV screen.

Let's examine some cost factors. While memory and logic costs continue to decrease, a 2 K byte programmable memory will always be cheaper than a 16 K byte one. To many designers, lower memory costs mean that they can design more memory into a system at the same price. To me it means that I can design lower cost computers using about the same amount I've always used. When the current crop of $1000 computers drops to $500, a COSMAC VIP type computer might drop to $150. The lower the cost, the more people can own their own computers.

The drastic price reduction for a central processing unit caused by the development of single chip microprocessors has not been matched by equivalent cost reductions for conventional computer input and output devices. The lowest cost home computers can only be achieved if the need for specialized IO devices is minimized. I don't think it is fair to ask the limited budget, unsophisticated beginner to spend several hundred dollars on a system which uses a $10 central processor but requires an additional expenditure of $1000 for a terminal and enough memory to make it useful. The COSMAC VIP minimizes IO device cost by relying only on a video monitor, audio cassette recorder and 16 position key pad. CMOS circuits reduce power supply costs, and a minimum number of integrated circuits permit the entire computer, including keyboard, to be packaged on one inexpensive printed circuit board.

As I see it, four major obstacles must be overcome before everyone can own their own computers. The first obstacle involves justifying a relatively high priced piece of equipment which is often useful to only one family member. COSMAC VIP was designed for relatively low cost and includes 20 video game listings that can be immediately loaded and played by the whole family. This goes a long way toward eliminating the first obstacle.

The second obstacle is the complexity of conventional computer operating and programming procedures. I can't understand how a beginner can be expected to cope with complex operating systems. COSMAC VIP programs are initiated with a single RUN switch. Any memory byte can be examined or modified using the built-in hexadecimal keyboard. Loading programs is easy. Just enter the starting address followed by the byte sequence to be stored. No extra key depressions are required between bytes. Memory addresses and stored bytes are displayed on the TV screen in hexadecimal format. Tapping a single key steps through memory letting the user examine stored bytes without modifying them.

To load a cassette program into memory, you merely enter the starting memory address and single digit block length code via the hexadecimal keyboard and play the tape. Less than 30 seconds later, COSMAC VIP shows you the last byte loaded into memory on the TV screen and you're ready to run your program. Recording the contents of memory on a cassette is just as easy.

A third obstacle in the path of home

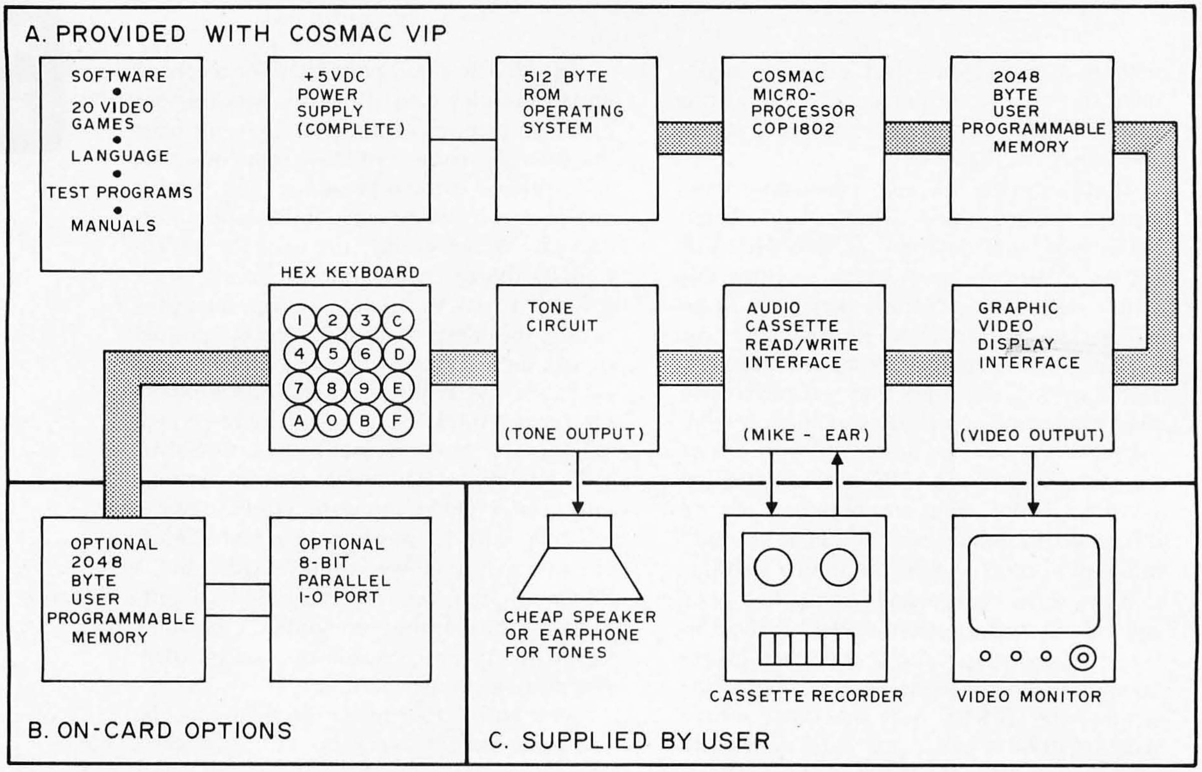


Figure 1: Block diagram depicting the essentials of a fully functioning COSMAC microprocessor. The system provides the user with everything needed to start programming except a video monitor and a cassette recorder.

computer purchase is the ability to program the computer. Suppose you buy an expensive system and then discover that you either can't or don't enjoy programming it. COSMAC VIP provides enough ready to use programs to justify the price even if you never write a program. It also provides a simple interpretive user language that many have found just as easy to learn as BASIC. This new language requires much less memory and is oriented toward simple graphics.

The fourth obstacle preventing many potential home users from getting started is the fear of buying an expensive computer and watching it become obsolete within several months. I feel that use of state of the art integrated circuits for COSMAC VIP helps minimize this problem, as does the relatively low cost of the system.

Hardware

Figure 1 shows a block diagram of the COSMAC VIP system. It shows exactly what is provided in the price and what the user has to provide. What you don't get is a video monitor, inexpensive speaker, standard audio cassette recorder and expensive cabinet. What you do get is everything else needed to immediately load and run 20 video games and write and debug your own graphic, game or educational programs. This includes all the memory you need and an easy to use numerically oriented language. You also have the ability to use COSMAC machine code.

All system timing is derived from a crystal clock so there are no adjustments required. A single tone can be generated by programs. This tone can be used in games or sounded when a key is pressed. The hexadecimal keyboard is a reliable touch pad type, and is fully debounced by read only memory software.

The audio cassette interface was designed to allow saving programs on tape. You load a program into memory from the hexadecimal keyboard and then record the contents on an audio cassette. You can then load the program from the audio cassette any time you want to use it again. The cassette data transfer rate is about 100 bytes per second. You must manually start and stop the cassette recorder, unless you add your own start and stop relays. The reliability of any audio cassette data system may not be suitable for commercial work. Audio cassettes are however the best means of saving programs available for the hobby computer user working on a limited budget. With proper care in use they have proven themselves to be quite adequate. In the COSMAC VIP tape system, each byte on

Continued on page 94

Circle 112 on inquiry card.

Our American Economic System is ☐ good ☐ bad. (check one)

We asked thousands of your fellow citizens this question. You should know how your answer compares with theirs.

Not too surprising, most people have gripes. Inflation. Unemployment. Taxes. These are among the concerns that topped the list.

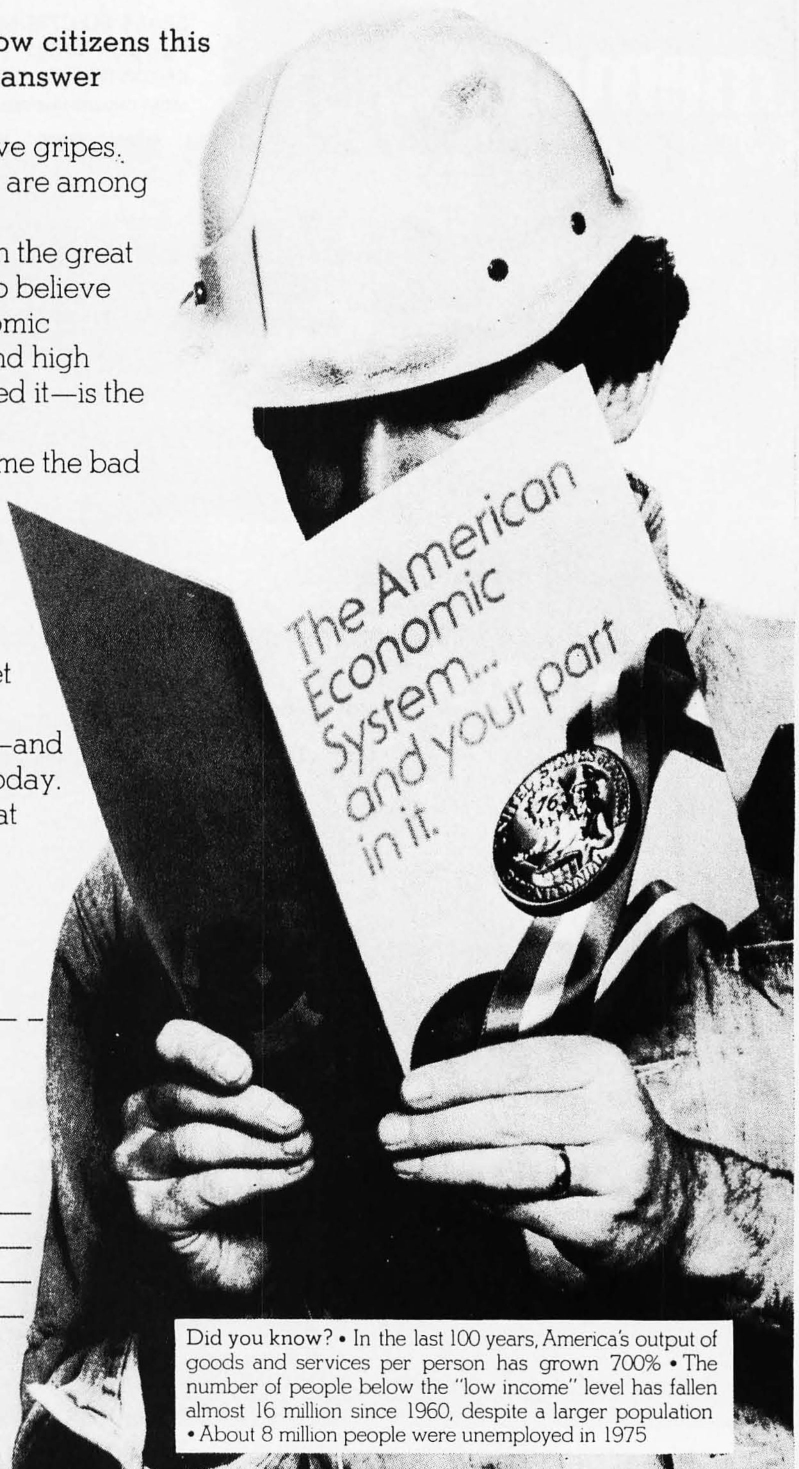
Yet, if you checked "good," you join the great majority of Americans (about 80%) who believe that fundamentally the American Economic System—with the individual freedom and high standard of living that have accompanied it—is the best in the world.

The question is: how do we overcome the bad without destroying the good?

Obviously, the more we all know about our system and how it works, the better we can decide what to preserve, what to change.

That's why we are offering a booklet that explains the American Economic System. It is easy to read, interesting—and free. Mail the coupon for your copy today.

Every American ought to know what it says.



The American Economic System.

It's one of your basic freedoms.

Booklets, P.O. Box 1887, New York, N.Y. 10001

__ Please send me a free copy of the booklet about our economic system.

__ I would also like a copy of the survey highlights.

Name ________________ Title ________

Company ________________________

Address ________________________

City __________ State ______ Zip ______

Note: Booklets in quantity, posters and other materials are available for use by companies, clubs, etc. For information, write: The Advertising Council, 825 Third Avenue, New York, New York 10022

A public service message of The Advertising Council & U.S. Department of Commerce presented by this Magazine.

Ad Council

Did you know? • In the last 100 years, America's output of goods and services per person has grown 700% • The number of people below the "low income" level has fallen almost 16 million since 1960, despite a larger population • About 8 million people were unemployed in 1975

Understanding APL

A Note to Those Unfamiliar with APL

If you are totally unfamiliar with APL, a good first step would be to consult some of the references listed at the end of this article and Mike Wimble's article. We recommend the IBM APL/1130 Primer, *available from IBM branch offices, as an excellent self-teaching tool*. . .CM

Dr Kenneth E Iverson
APL Design Group
IBM
1700 Market St
Philadelphia PA 19103

Note: In the APL listings which follow, the user's entries are reproduced in blue while the computer's responses are in red for clarity.

Adin Falkoff, in his review of APL conferences to be published in *Computer Reviews* (as of this writing), indicates the "widely ranging subject matter" covered in APL papers by listing dozens of applications, covering such areas as word processing, convex geometry and database systems, to name a few. Because of this potential breadth, it is necessary, in a short paper, to focus more narrowly on some one aspect of APL.

In scanning previous issues of BYTE to identify some aspect which would be relevant to its readers, I concluded that, however diverse their interests, this audience held one thing in common: a penchant for experimentation and exploration, for playing around with ideas and things. I have therefore chosen to emphasize these aspects of APL: the use of experimentation on an APL computer in learning the language itself, and the exploration of other topics both with and without the aid of a computer.

True experimentation must be carried out on a real physical device. Readers who have the use of APL computers or terminals may find them useful for verification of the questions posed later in the article. For other readers we will try to approximate this experience by showing certain experiments together with the results they produce. The reader may then study the results of these experiments in the same way he or she would study the results of his or her own experiments on a real computer.

However, unlike the experimenter at a real APL computer, the reader cannot then decide what experiment to try next in order to settle the questions raised in his or her mind by the last experiment, but can only use the experiments next presented by the author. Although each experiment may have been chosen to answer questions raised by its predecessors, it may miss many of the questions raised in the minds of particular readers. However, this difficulty can also be overcome to a large degree by the use of "thought experiments," discussed later in this article.

To fully appreciate and enjoy the experimental approach used here, one should make the most of each experiment by *not* reading any result until every clue has been used to puzzle out what the result should be. For example, one would try to reason out and state the definition of the function ⌈ from the following experiments:

```
      3⌈5                    5⌈3
5                      5
      4⌈7                    ¯5 ⌈3
7                      3
```

before going on to read that "the symbol ⌈ denotes the maximum function which yields the value of the larger of its arguments." Remember, of course, that APL statements (shown in red here) start at the left margin, while user entries are indented. Similarly, try to guess the result of the following experiment:

```
8    12⌊8
```

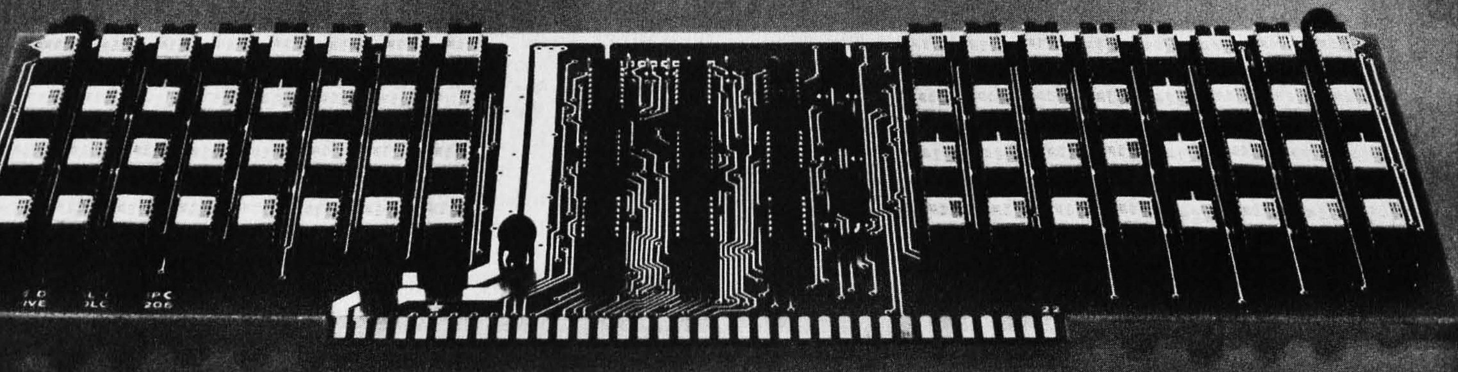
before reading it, since the relation of the symbols ⌈ and ⌊ alone suggests that ⌊ denotes the minimum function.

One final comment on the reading of this paper: although it is clear that this magazine enjoys the patronage of many young readers, I have, in the interests of clarity and precision, chosen to use words which may be unfamiliar to some readers. If you happen to fall into this class, then this is as good a time as any to acquire the important habit of reading with a good dictionary at your elbow.

Systematic Experimentation

Simple functions yield to experiments with randomly chosen arguments, as happened in the case of the maximum function. Even there, however, the arguments were chosen with more care than might appear. For example, if only the two on the left (3⌈5 and 4⌈7) had been given, one might

Memories are made of this.



32K. One Card. One low price. Only from the Digital Group.

Now, on only one *fully static* card, the Digital Group has squeezed in a whopping 32K of memory. Which, with a little quick addition, means a full 64K architecture now requires only 2 boards instead of 8. That's a 4-to-1 space reduction ...and leaves one extra memory slot on the Digital Group's standard motherboard still available for future products.

All this and one low price, too.
It just may be the best news of all. Our full static, assembled and tested 32K memory board is only $995. Now that's worth remembering. It's substantially less than our equivalent assembled 8K board prices. (Please note: We're initially offering this 32K board assembled only, but kit versions will soon be available, too—at even lower prices.)

Here's what you get.

Specifications:

- 32K on single card
- Speed—450ns. All of our current CPUs will operate at full-rated speed.
- Decoding—Lower or upper 32K bank
- Power— +5V only @ 4A
- Card size—12" x 5" (excluding connector fingers)

Features:

- May be intermixed on Digital Group systems with our 8K memory cards
- All data and address lines are buffered
- *Fully* static memories—EMM 4801 (450ns) or equivalent

Price:

32K board complete, assembled and tested $995.00

For all the memorable details, just fill out the coupon below. (Then all you have to remember is to mail it in.)

---

the digital group

P.O. Box 6528
Denver, Colorado 80206 (303) 777-7133

☐ **I promise to mail this in, so add me to your mailing list!**

Name ______________________

Address ______________________

City/State/Zip ______________________

☐ Remember me? I'm already on your mailing list, but I need the memory spec sheet desperately.

Circle 114 on inquiry card.

have concluded that the result of the function was simply the value of its right argument, but the third case (5⌈3) ruled out this possibility.

Experimentation with a systematic set of arguments usually gives a clearer picture of the function. For example, we might use a fixed left argument and a set of successive integers for the right argument, as in 3⌈1 and 3⌈2, and so on. This can be done most conveniently by giving a list of arguments as follows:

```
      3×1 2 3 4 5 6 7
3 6 9 12 15 18 21
      3+1 2 3 4 5 6 7
4 5 6 7 8 9 10
      3⌈1 2 3 4 5 6 7
3 3 3 4 5 6 7
      3≥1 2 3 4 5 6 7
1 1 1 0 0 0 0
      3≠1 2 3 4 5 6 7
1 1 1 0 1 1 1
```

The behavior of the two new functions (≥ and ≠) are probably clear from the last two experiments. However, the experiment:

```
      3|1 2 3 4 5 6 7
1 2 0 1 2 0 1
```

might leave one in doubt, wishing to see cases with different left arguments such as 4, 5, etc. In general, one might wish to see a whole table of values for a systematic set of left arguments. This will be explored in the section on tables.

Names

It is clearly convenient to give a name to any number or list or table of numbers that is used repeatedly. For example, we could give the name *S* to the list 1 2 3 4 5 6 7 as follows:

```
      S←1 2 3 4 5 6 7
```

We could then write the preceding experiments more simply as 3x*S* and 3+*S* and 3⌈*S*, etc. Thus:

```
      3×S
3 6 9 12 15 18 21
      2*S
2 4 8 16 32 64 128
      3*S
3 9 27 81 243 729 2187
      S*2
1 4 9 16 25 36 49
      S*3
1 8 27 64 125 216 343
```

This use of names corresponds to the notion of *variables* in algebra.

Tables

In the experiment:

```
      4×S
4 8 12 16 20 24 28
```

it is clear that the multiplication applies to the left argument 4 together with each element of the right argument. In the experiments:

```
      S+S
2 4 6 8 10 12 14
      S×S
1 4 9 16 25 36 49
```

it is clear that the functions apply to each of the corresponding pairs of elements of the left and right arguments.

In order to make an addition table for the integers from 1 to 7, it would be necessary to add each element of the left argument to every element of the right argument. This will be indicated by placing the symbols for null (∘) and dot (.) before the plus sign as follows:

```
      S∘.+S
2  3  4  5  6  7  8
3  4  5  6  7  8  9
4  5  6  7  8  9 10
5  6  7  8  9 10 11
6  7  8  9 10 11 12
7  8  9 10 11 12 13
8  9 10 11 12 13 14
```

Tables for times (x), maximum (⌈), and less than (<) can be produced similarly:

```
      S∘.×S              S∘.⌈S            S∘.<S
1  2  3  4  5  6  7 | 1 2 3 4 5 6 7 | 0 1 1 1 1 1 1
2  4  6  8 10 12 14 | 2 2 3 4 5 6 7 | 0 0 1 1 1 1 1
3  6  9 12 15 18 21 | 3 3 3 4 5 6 7 | 0 0 0 1 1 1 1
4  8 12 16 20 24 28 | 4 4 4 4 5 6 7 | 0 0 0 0 1 1 1
5 10 15 20 25 30 35 | 5 5 5 5 5 6 7 | 0 0 0 0 0 1 1
6 12 18 24 30 36 42 | 6 6 6 6 6 6 7 | 0 0 0 0 0 0 1
7 14 21 28 35 42 49 | 7 7 7 7 7 7 7 | 0 0 0 0 0 0 0
```

The patterns of numbers (such as the diagonals of twos, threes, etc, in the addition table, and the L-shaped patterns in the maximum table) are often surprising even if one is already familiar with the particular function, and it is entertaining to predict patterns before actually producing the table.

Tables are also useful in determining the definition of an unknown function. To illustrate this we will return to the function | for which we have already seen the case:

```
      3|1 2 3 4 5 6 7
1 2 0 1 2 0 1
```

The following table should suffice to identify the function for most readers:

```
      S∘.|S
0 0 0 0 0 0 0
1 0 1 0 1 0 1
1 2 0 1 2 0 1
1 2 3 0 1 2 3
1 2 3 4 0 1 2
1 2 3 4 5 0 1
1 2 3 4 5 6 0
```

Some Tools

The functions introduced by the following experiments are not only interesting

Here's everything you need to program your own '8080'!



SCELBI does it again! Here, for the first time, are all the tools you need to get the most out of your '8080'. About the only thing missing is the '8080' itself. Just look at what's in this total package . . .

'8080' Octal Code Card and/or **'8080' Hexadecimal Code Card,** instant slide rule aids for programming or debugging '8080' software. Contains all standard mnemonics, with corresponding codes for either octal or hexadecimal modes. Easy-reading instructions are color coded to indicate which flags are affected during instruction execution. Front of card features quick, logical reference formats, while back displays ASCII code chart for all 128 characters, plus '8080' status words and register pair codes.

The '8080' Programmer's Pocket Guide is a compact 3 x 4½" ever-ready, instant reference for either Code Card, that explains the instruction set in detail, giving both octal and hexadecimal codes.

SPECIAL MONEY SAVING INTRODUCTORY OFFER!

Each of these "must" programming tools sells individually for only \$2.95. If you order today, you can own any two for only \$5.50 . . . or all three for only \$7.95! You need these low cost, highly productive tools. So, order today, and save, Save, SAVE! Clip this handy mail order coupon and mail it today!

Scelbi Computer Consulting, Inc.
P.O. Box 133 B PP STN
Milford, CT 06460

Please rush items checked below (Write quantities of each in ◯)

◯ **Octal Code Card(s)** ◯ **Hexademical Code Card(s)**

◯ **The '8080' Programmer's Pocket Guide(s)**

- ☐ I enclose \$2.95 for item indicated above.
- ☐ I enclose \$5.50 for 2 items indicated above.
- ☐ I enclose \$7.95 for 3 items indicated above.

LIMIT: 3 items per customer for this money-saving Introductory Offer.

Name ____________________

Address ____________________

City ____________ State ______ Zip ______

Circle 115 on inquiry card.

in themselves, but prove very useful in further experiments:

```
      +/3 2 7 5
17
      3+2+7+5
17
      ×/3 2 7 5
210
      ⌈/3 2 7 5
7

      T←(⍳4)∘.-⍳4
0 ¯1 ¯2 ¯3
1  0 ¯1 ¯2
2  1  0 ¯1
3  2  1  0
```

```
      ⍳3
1 2 3
      ⍳4
1 2 3 4
      5⍴3
3 3 3 3 3
      4⍴2
2 2 2 2

      ⍉T
 0  1  2  3
¯1  0  1  2
¯2 ¯1  0  1
¯3 ¯2 ¯1  0
```

```
      ⌽3 2 7 5
5 7 2 3
      ⌽⍳5
5 4 3 2 1
      1 0 1 0 1/⍳5
1 3 5
      0 1 0 1 0/⍳5
2 4

      +/T
¯6 ¯2 2 6
      +/⍉T
6 2 ¯2 ¯6
      ⌈/T
0 1 2 3
```

These functions can be used in many interesting ways to bring out the properties of other functions. For example, the relation between addition and multiplication and the relation between multiplication and power can be brought out by considering the following expressions for various values of *N* and *M*:

```
      N←4
      M←3
      +/N⍴M
12
      M×N
12
      ×/N⍴M
81
      M*N
81
```

```
      N←5
      M←2
      +/N⍴M
10
      M×N
10
      ×/N⍴M
32
      M*N
32
```

Thought Experiments

Thus far we have considered only experiments performed with the aid of a computer or, what amounts to the same thing, with the aid of printed results prepared in advance by a computer. However, if the rules governing the notation are simple and uniform (as they are in APL), then one can perform interesting and revealing *thought experiments* by simply applying the rules. For example, the following expressions can be seen to be equivalent by performing thought experiments for various chosen values of *N*:

```
+/⍳N
+/⌽⍳N
((+/⍳N)+(+/⌽⍳N))÷2
(+/((⍳N)+⌽⍳N))÷2
(+/((N⍴(N+1))))÷2
(N×(N+1))÷2
```

In particular, the last expression is a neat way of evaluating the first. For example, if *N* is 100, then +/⍳*N* is the sum of the first 100 integers, and is equal to (100x101) ÷ 2, or 5050.

We will conclude with two thought experiments which use only functions already treated. The first:

```
(2=+/⍉0=(⍳N)∘.|⍳N)/⍳N    *
```

has a result which, when obtained for any argument *N* equal to 7 or more, should be recognized by any reader.

** As a challenge to readers, we will print the answer to this thought experiment next month together with a complete explanation of how the expression is evaluated.*

The second concerns comparisons of two tables *M* and *T*. The expression *M*≠*T* yields a table of the same size containing ones and zeros, with ones only where *M* and *T* disagree. Therefore +/*M*≠*T* gives counts of the disagreements in each row, and +/+/*M*≠*T* gives a single total count. Hence the latter expression yields zero only if *M* and *T* are identical.

The expression +/+/*M*≠⍉*M* therefore yields zero only if *M* agrees with its transpose ⍉*M* obtained by "flipping *M* about its diagonal to interchange the rows with the columns." Evaluate the expression +/+/*M*≠⍉*M* for values of *M* produced as function tables for various functions such as:

```
M←S∘.+S     M←S∘.×S     M←S∘.-S     M←S∘.≤S
```

Then try to determine what class of functions yields a zero result.

Ideas for experiments in a variety of fields are available in existing publications. Among the references cited at the end of this article, Iverson (references 2 and 3) presents further definitions of APL functions, Blaauw (4) presents a 1 page APL definition of the Intel 8080 instruction set, the exercises in Iverson (5 and 6) and in Orth (7) present topics in mathematics at various levels, Spence (8) treats electric circuits, and Iverson (9) treats a variety of topics which include geometry and computer science.■

REFERENCES

1. Falkoff, A D, *Computing Reviews* (to appear).
2. Iverson, K E, *An Introduction to APL for Scientists and Engineers*, APL Press, Pleasantville NY, 1976.
3. Iverson, K E, *Introducing APL to Teachers,* APL Press, Pleasantville NY, 1976.
4. Blaauw, G A, *Digital System Implementation*, Prentice Hall, Englewood Cliffs NJ, 1976.
5. Iverson, K E, *Algebra: an algorithmic treatment*, APL Press, Pleasantville NY, 1972.
6. Iverson, K E, *Elementary Analysis*, APL Press, Pleasantville NY, 1976.
7. Orth, D L, *Calculus in a new key*, APL Press, Pleasantville NY, 1976.
8. Spence, R, *Resistive Circuit Theory*, McGraw-Hill, New York, 1974.
9. Iverson, K E, *APL in Exposition*, APL Press, Pleasantville NY, 1976.
10. *American Heritage Dictionary*, American Heritage Publishing Co, 1969.

IF YOU'RE NOT DESIGNING WITH A CSC PROTO-BOARD,® LOOK AT ALL YOU'RE MISSING.

Accessibility—All parts are instantly and easily accessible, for quick signal tracing, circuit modifications, etc.

Utility—Models are available with or without built-in regulated power supplies (fixed or adjustable).

Variety—A wide variety of models are available with capacities ranging from 630 to 3060 solderless tie-points (6 to 32 14-pin DIP's), to fit every technical and budget requirement.

Economy—Eliminate heat and mechanical damage to expensive parts. Save money by re-using components.

Versatility—Use with virtually all types of parts, including resistors, capacitors, transistors, DIP's, TO-5's, LED's, transformers, relays, pots, etc. Most plug in directly, in seconds.

Durability—All Proto-Board models are carefully constructed of premium materials, designed and tested for long, trouble-free service.

Expandability—Proto-Board units can be instantly inter-connected for greater capacity.

Visibility—All parts are instantly and easily visible, for quick circuit analysis and diagramming.

Speed—Assemble, test and modify circuits as fast as you can push in or pull out a lead. Save hours on every project.



Adaptability—Use in design, packaging, inspection, QC, etc. Works with most types of circuits, in many, many applications.

Flexibility—Use independently, or in conjunction with other accessories, such as scopes, counters, CSC Proto-Clip™ connectors, Design Mate™ test equipment, etc. One Proto-Board unit can serve a thousand applications.

Whatever type of electronic circuits you work with, you can do more in less time with CSC's solderless Proto-Board systems. As fast and easy as pushing in or pulling out a lead, you can design, test and modify circuits at will. Components plug into rugged 5-point terminals, and jumpers, where needed, are lengths of #22 AWG solid wire. In the same time you took to read this ad, you could be well on your way to assembling a new circuit. For more information, see your CSC dealer, or write for our catalog and distributor list.

CSC PROTO-BOARD SOLDERLESS BREADBOARDS

MODEL NUMBER	NO. OF SOLDERLESS TIE-POINTS	IC CAPACITY (14-PIN DIP'S)	MANUFACTURER'S SUGG LIST	OTHER FEATURES
PB-6	630	6	$15.95	Kit—10-minute assembly
PB-100	760	10	19.95	Kit—with larger capacity
PB-101	940	10	29.95	8 distribution buses, higher capacity
PB-102	1240	12	39.95	Large capacity, moderate price
PB-103	2250	24	59.95	Even larger capacity; only 2.7¢ per tie-point
PB-104	3060	32	79.95	Largest capacity; lowest price per tie-point
PB-203	2250	24	75.00	Built-in 1%-regulated 5V, 1A low-ripple power supply
PB-203A	2250	24	120.00	As above plus separate ½-amp +15V and −15V internally adjustable regulated power supplies

CONTINENTAL SPECIALTIES CORPORATION

44 Kendall Street, Box 1942, New Haven, CT 06509
203-624-3103 TWX 710-465-1227
West Coast: 351 California St., San Francisco, CA 94104
415-421-8872 TWX 910-372-7992

Prices and specifications subject to change without notice.

Circle 116 on inquiry card.

The Complete Challenger System



Imagine a system complete with terminal, CPU, memory, floppy disk, software, and all the little necessities to make it work together immediately. Now imagine this complete system available not only fully assembled, but priced much lower than anyone else's kit. What you are dreaming of is OSI's "new" Challenger System!

In the configuration shown above, the Challenger includes everything an end user needs for a complete small computer system. All you add is 110 VAC power and a desk to put it on.

This fully-assembled system includes:

HARDWARE:

OSI Challenger 65 with 16K RAM, serial interface, system monitor PROM, and floppy disk bootstrap PROM.

OSI Challenger single drive floppy disk formatted for 250K bytes storage per diskette surface.

Stand-alone terminal and Sanyo monitor for 16 lines of 64 characters at 2400 baud (other terminal options are available).

And all interconnecting cables!

SOFTWARE:

2 diskettes containing over 100,000 bytes of software including OSI's powerful Disk Operating System with variable length sectors. **6502 DISK BASED RESIDENT ASSEMBLER/ EDITOR!** A totally interactive Assembler/Editor which assembles up to 600 lines a minute and is completely compatible with MOS Technology's Cross Assembler format. This program also contains a powerful disk-based line editor with commands for general text editing. **OSI'S EXTENDED MONITOR:** A powerful machine language debugging and utilities package including a Disassembler which is format compatible with the Assembler! **OSI 6502 8K BASIC FOR DISK BY MICROSOFT:** This powerful BASIC has all the features of Altair® 8K BASIC for the 8080 plus higher speed and disk storage. And it comes complete with a BASIC program library.

DOCUMENTATION AND SUPPORT:

We include over 600 pages of hardware, software, programming, and operation manuals. The Challenger is based on the well-proven OSI 400 system. The over 2,000 OSI 400s and Challengers now in use assure continuing hardware and software support for this system for years to come!

EXPANDABILITY:

The Challenger System can now be expanded to 192K of RAM and 16K of I/O and ROM. There are over 13 accessory boards including A/D, D/A, parallel and serial I/O, cassette interfaces, a dual drive floppy, a video graphics display, several RAM and PROM boards, and multiple-processor configurations.

APPLICATIONS:

The Challenger system is complete, fully assembled and configured so that the Disk Operating System can be booted in immediately on system power-up. Even a relatively inexperienced operator can have a complex BASIC program on-line just seconds after the system is turned on. The ease of use, high reliability, and large library of standard BASIC applications programs make the OSI Challenger System the first practical and affordable small computer system for small business, educational institutions, labs, and the personal computerist.

PRICES:

Challenger System, complete as stated above with terminal and monitor

$2599.00

As above without terminal. Specify RS-232 or 20ma loop and baud rate

$2099.00

IMPORTANT NOTE:

One of the most important features of the Challenger System is that it is not really "new". OSI has been delivering the basic circuitry of the Challenger since November 1975 and the floppy disk since June 1976. The only thing new is the total integration of the components as a complete, simple to use, fully-assembled, small computer system.

For more free information and the address of the OSI Computer Dealer or representative in your area, write to: OSI; Dept. S; Hiram, Ohio 44234 or enclose $1.00 for the full OSI catalog which contains kits from $134 and fully assembled computers from $439.

OHIO SCIENTIFIC

Dept. S, Hiram, Ohio 44234

Listing 1: An APL Lunar Lander program. The program consists of the main program function FLY, plus an optional initialization program LEM and a set of subfunctions called from FLY. All input and output operations are contained in the subfunctions so that the operator can more easily experiment with different output formats. The program has been written for a hypothetical APL interpreter designed for use in microcomputers, and as such uses a subset of the standard APL operators. Note the compact nature of the program (an APL hallmark), and the fact that line numbers used in program functions can be repeated in subfunctions.

```
        ∇ FLY
[1]  L550:SB←Z←B←3ρT3←ST←T←0
[2]   E←3ρ10
[3]   F←F3
[4]   D←A3,D4,D5
[5]   V←V6,V4,V5
[6]  L640:TF←|(((|1↑(V×V)-2×D×G)*0.5)+1↑V)÷1↑G
[7]   CKSCALE
[8]   PTSTAT
[9]   T←⍎'T',(0.01>F)/'F'
[10]  →(0.01>F)/L830
[11]  SETVALUE
[12]  T←MT⌈1↑(|T)⌊⌈TF
[13]  Z←Z+M<|B
[14]  B←(M⌊|B)××B
[15]  B←((|B)⌊(0⌈F-+\¯3↑2↑B×T)÷T)××B
[16]  →(0.01≤F←0⌈F-T×+/|B)/L1530
[17]  '***** OUT OF FUEL *****'
[18] L1530:→(∨/(0=E),2=Z)/L1780
[19]  (Z≠0)/[0](3 10 ρ'VERTICAL  TRANSVERSEHORIZONTAL'),(3 21 ρ' ENGIN
      E WILL BLOW IN '),(3 3 ρ 3 0 ⍕E), 3 7 ρ' BURNS.'
[20]  E←E-Z≠0
[21] L830:D←D+(V×T)+(T×T×G+B)÷2
[22]  V←V+T×G+B
[23]  B←B×0.01<F
[24]  →(0.001<1↑D)/L640
[25]  ENDGAME
[26]  →0
[27] L1780:'******BANG!'
    ∇
        ∇ CKSCALE;DS1;DS2;PLOT
[1]   →(1000≤1↑D)/0
[2]   →(100≤1↑D)/L1
[3]   '******PREPARE FOR IMPACT'
[4]   →(10≤1↑D)/L1
[5]   '**********FASTEN RETAINING STRAPS'
[6]  L1:DS1←¯1↑(0 20 100 500 2500 <|1↑1↓D)/ 4 20 100 500 2500
[7]   DS2←¯1↑(0 20 100 500 2500 <|¯1↑D)/ 4 20 100 500 2500
[8]   'SCALE:X=',(⍕DS2),',Y=',⍕DS1
[9]   PLOT←121ρ0
[10]  PLOT[5+11×⍳11]←6
[11]  PLOT[55+⍳11]←11ρ5
[12]  PLOT[60+(11×(5⌊⌊|D[1]÷DS1)××D[1])+((5⌊⌊|D[2]÷DS2)××D[2])]←1
[13]  PLOT[60]←2+PLOT[60]
[14]  ⌽ 11 11 ρ⌽' ÷⎕⌹+|-⎕⎕'[PLOT]
    ∇
        ∇ PTSTAT
[1]   'AT TIME T = ',(⍕T3←T3+T),' SEC. STATS ARE AS FOLLOWS:'
[2]   '           VERTICAL           TRANVERSE           HORIZONTAL'
[3]   'DISTANCE',9↓(18 2 ⍕D),' METERS'
[4]   'VELOCITY',9↓(18 2 ⍕V),'MTRS/SEC.'
[5]   'FUEL... ',(2⍕F),' BURN UNITS'
[6]   'COMPUTED TIME TO IMPACT AT PRESENT RATE... ',(2⍕TF×D[0]>
      0.001),' SECONDS.'
    ∇

      )LOAD LEM
SAVED  18.18.40 01/03/77
THIS IS THE "LUNAR LANDER" PROGRAM
TYPE "LEM" TO ENTER A NEW SET OF SPACE AND VEHICLE CONDITIONS
TYPE "FLY" TO USE THE PRESTORED SET OR TO RETRY THE SET YOU ENTERED IN "LEM".
        ∇ LEM
[1]   'THIS IS A LUNAR LANDING PROGRAM'
[2]   'ENTER INITIAL ALTITUDE IN METERS...'
[3]   A3←⎕
[4]   'ENTER INITIAL VERTICAL VELOCITY (DOWN IS MINUS)...'
[5]   V6←⎕
[6]   'ENTER TRANSVERSE DISTANCE (Y)...'
[7]   D4←⎕
[8]   'ENTER Y VELOCITY...'
[9]   V4←⎕
[10]  'ENTER HORIZONTAL (X) DISTANCE...'
[11]  D5←⎕
[12]  'ENTER X VELOCITY...'
[13]  V5←⎕
[14]  'ENTER MAXIMUM BURN RATE...'
[15]  M←⎕
[16]  'ENTER FUEL CAPACITY...'
[17]  F3←⎕
[18]  'ENTER PLANETARY FORCES (GRAVITY, Y AND X CORIOLIS COMPONENTS)'
[19]  G←3ρ⎕,0,0,0
[20]  'ENTER MINIMUM SPECIFIABLE TIME DELTA...'
[21]  MT←⎕
[22]  'ENTER NAME OF YOUR SHIP...'
[23]  N←⍞
[24]  ' '
[25]  ' '
[26]  '***CONTROL TO ',N,'... COMMENCE LANDING'
[27]  FLY
    ∇
```

Here's APL

David D Keefe
144A Spring St
Tillson NY 12486

About the Game

This APL version of the Lunar Lander program is developed from a BASIC version called LEM, published in the November-December 1976 issue of *Creative Computing* magazine ("LEM" contributed by Bill Cotter). The program has been restructured

The APL System

This APL Lunar Lander program has been written using a significant subset of the full APL operator set (such as APLSV implemented on the IBM 5100), and is eventually intended for use on a small system APL interpreter. In general, the functions use no array operations or vector subscripting (indexing). However, for illustration purposes specific examples of array reduction and vector indexing are included. In these cases the text describes alternate procedures with examples to show why it is advantageous to keep this higher level of capability. The operator subset used is based on a proposal contained in Mark Arnold's article "What is APL?" (November 1976 BYTE, page 20), which has been expanded slightly to include the execute and vector rotation operators.

About the Author: *Dave Keefe describes himself as an incredibly lazy programmer. He is convinced that APL lets him write, test and bring up a new application in less time, less effort and with less symbols than any other major language. He wrote this article over the holiday season in the hope that the game format would make a good vehicle for showing how APL can be applied. The game program took a couple of evenings at an APL machine's terminal. The rest of the article took considerably longer.*

in Action!

to use vector operations within the force equations, resulting in a simplified program structure. Vector operations also allow the game to be easily expanded: For example, the user can now specify Coriolis force elements, seen as an effect on the horizontal and transverse velocities as a result of the rotation of the target planet. These forces are applied as a part of the same equation which applies the gravitational force.

The Program

The APL LEM program consists of a main program function (FLY), an optional initialization function (LEM), and a set of subfunctions which are called from within FLY. In general, the input and output operations associated with LEM have been put into separate functions. This permits experimentation with variations of the IO formats without affecting the main processing program which calls these functions. The accompanying write-up describes each of the statements in the FLY program and the general operations performed in the other functions. It is intended as a tutorial to be read in conjunction with the function listings, which continue on the next two pages.

Listing 1, continued:

```
        ∇ SETVALUE
[1]    'T' SET 'ENTER TIME INTERVAL-  '
[2]    'B[0]' SET '     VERTICAL (Z) BURN?  '
[3]    'B[1]' SET '     TRANSVERSE (Y) BURN? '
[4]    'B[2]' SET '     HORIZONTAL (X) BURN? '
    ∇
        ∇ A SET MSG;V
[1]    →(1=⍎'S',A)/0
[2]    ⍞←MSG
[3]    V←(⍴MSG)↓⍞
[4]    →(∨/V='→')/SS
[5]    ⍎A,'←FIX V'
[6]    →0
[7]   SS:⍎'S',A,'←1'
[8]    ⍎A,'←0'
[9]    →0
    ∇
        ∇ Z←FIX D
[1]    D[1↓Z/⍳⍴Z←D='.']←' '
[2]    Z←(D≠' ')/D
[3]    D←1↑Z,'0'
[4]    Z←(0≠⍴Z)/((~D∊'.0123456789')/'¯'),Z←(Z∊'.0123456789')/Z
[5]    Z←1↑⍎Z,' ',⍕⍎A
    ∇
        ∇ ENDGAME
[1]    D[0]←0
[2]    CKSCALE
[3]    PTSTAT
[4]    →(∧/3≥|V)/SUCC
[5]    (8 8 ⍴'OH-OH...SHUCKS..SORRY...AAAAARGHWHOOPS..⍴⌈?>≠!!!AIEEEEEEGDHELPUS')[
       ¯1++/ 1 2 4 ×Z←3<|V+G;]
[6]    ' '
[7]    '******AT ',(0⍕T3÷60),' MIN., ',(2⍕60|T3),' SEC. INTO THE LANDING PROCEDURE'
[8]    '      ALL RADIO CONTACT WITH THE ',N,' WAS LOST.'
[9]    '      AT LAST REPORT SHE WAS ',((0.01>F)/'OUT OF FUEL AND '),'DESCENDING AT THE R
       ATE'
[10]   '      OF ',(2⍕V[0]),' METERS/SEC., WITH A GROUND SPEED OF ',(2⍕B←(+/B×B←1↓V+G)*
       0.5),' METERS/SEC.'
[11]   '******HER LAST REPORTED POSITION WAS ',(2⍕(+/D×D←1↓D)*0.5),' METERS FROM THE TARG
       ET LANDING SITE.'
[12]   '      SHE IS PRESUMED TO HAVE ',((3<|V[0])/'CRASHED ',(3<B)/' AND '),((3<B)/'FLIP
       PED OVER '),'UPON IMPACT.'
[13]   →0
[14] SUCC:'******THE ',N,' HAS LANDED!******
       '
[15]   →(∧/4>|B←(+/B×B←1↓D)*0.5)/GRT
[16]   →(∧/20>|B)/OK
[17]   →(∧/100>|B)/FAIR
[18]   →(∧/500>|B)/POOR
[19]   'OF COURSE YOU ARE HOPELESSLY LOST IN THE WILDERNESS.'
[20]   →0
[21] FAIR:'TOO BAD THAT YOU ARE BALANCED ON THE EDGE OF A CREVASSE... YOU WERE ',(
       7 2 ⍕B),'METERS OFF TARGET'
[22]   →0
[23] OK:'YOU ARE ALMOST READY TO SOLO WITH THAT PERFORMANCE!'
[24]   →0
[25] GRT:'YOU QUALIFY FOR SENIOR PILOT WITH THAT ONE!'
[26]   →0
[27] POOR:'UNFORTUNATELY YOU ARE IN THE MIDDLE OF A RATHER DEEP RIVER
                       WORSE YET, YOU ARE 3 METERS UPSTREAM OF A 1000 METER WATERFALL!'
[28]   →0
    ∇
```

Listing 2: An explanation of the Lunar Lander program shown in listing 1. Author Keefe provides detailed comments about the program, a virtual necessity when programming or reviewing programs in APL, because of its conciseness. In the case of the subfunction CKSCALE, an alternate program is described which avoids the use of vector and array indices.

```
FUNCTION NAME: FLY

     THIS FUNCTION IS THE PRINCIPAL CALCULATION AND LOGIC
     PROCESS IN THE LUNAR LANDER (ALIAS - THE MAIN LINE). BRANCH
     POINTS WITHIN FLY ARE LABELLED "Lxxx" WHERE xxx IS THE LINE
     NUMBER IN THE BASIC VERSION OF LEM WHICH IS MOST COMPARABLE
     TO THE PROCESSING WHICH OCCURS AT THIS STATEMENT. THE
     FOLLOWING SECTION PROVIDES A LINE BY LINE DESCRIPTION OF
     THE PROCESSING WHICH IS PERFORMED IN FLY:

LINE NO|     DESCRIPTION
-------+--------------------------------------------------------
      1 THIS LINE SHOWS HOW A SINGLE APL STATEMENT CAN BE USED TO
              SPECIFY MULTIPLE VARIABLES FOR BOTH VECTORS AND
              SCALARS. THIS STATEMENT SETS ALL OF THE
              VARIABLES NAMED TO ZERO. SB, Z, AND B ARE 3
              ELEMENT VECTORS; T3, ST, AND T ARE SCALARS.

      2 E IS A VECTOR WHICH SHOWS THE NUMBER OF BURNS LEFT FOR A
              DAMAGED ENGINE. IF ANY ELEMENT OF E EVER REACHES
              ZERO THE ENGINE WILL EXPLODE.

      3 F IS THE REMAINING FUEL. IT IS INITIALIZED TO THE VALUE
              INPUT IN THE LEM ROUTINE.

      4 D IS THE DISTANCE VECTOR CONTAINING THE ALTITUDE,
              TRANSVERSE AND HORIZONTAL DISTANCE FROM THE
              TARGET LANDING AREA. NOTE HOW CATENATION IS USED
              TO FORM THE VECTOR FROM THE THREE SCALARS WHICH
              WERE ENTERED IN LEM .

      5 V IS THE VELOCITY VECTOR MADE UP OF THE VERTICAL,
              TRANSVERSE AND HORIZONTAL VELOCITIES BEING
              APPLIED TO THE LANDER.

      6 L640 - THE START OF THE MAIN PROCESSING LOOP. TF IS THE
              CALCULATED TIME UNTIL IMPACT ASSUMING THE
              VERTICAL ENGINE IS SHUT DOWN. THIS IS SIMPLY A
              SOLUTION FOR THE POSITIVE REAL ROOT OF THE
              QUADRATIC EQUATION [0=D+(V×TF)+(.5×G)×TF*2].

      7 THE CKSCALE FUNCTION PRINTS THE GRID SHOWING LEM LOCATION
              VERSUS THE TARGET AS THE LEM APPROACHES
              TOUCHDOWN.

      8 THE PTSTAT FUNCTION PRINTS THE CURRENT FLIGHT STATISTICS.

      9 AND 10 PROVIDE A SHORTCUT FOR THE CASE WHERE THE LEM HAS
              RUN OUT OF FUEL. STATEMENT 9 FORCES THE TIME TO
              THE FULL TIME-TO-IMPACT AND 10 THEN BRANCHES
              DIRECTLY TO THE MAIN CALCULATION SEGMENT.
              THESE STATEMENTS COULD HAVE BEEN COMBINED INTO A
              SINGLE COMPOUND STATEMENT. FOR EXAMPLE,
              →(FT,0⍴T←⍎'T',(FT←0.01>F)/'F')/L830 HOWEVER THE
              CATENATION FT,0⍴T←... IS ARTIFICIAL AND NOT
              CONSIDERED GOOD FORM IN APL.

     11 THE SETVALUE FUNCTION READS IN NEW VALUES FOR THE TIME
              INTERVAL AND BURN RATE (T AND B).

     12 THE RESULTANT VALUE OF T WILL BE EQUAL OR GREATER THAN
              THE MINIMUM ALLOWED TIME (MT) AND NO GREATER THAN
              THE NEAREST WHOLE SECOND GREATER THAN THE
              ZERO-BURN TIME TO IMPACT (TF).
```

Listing 2, continued:

```
13 A REAL EXAMPLE OF THE POWER OF APL. NOTE HOW THE RESULTS
        OF THE LOGICAL OPERATION M<|B (MAXIMUM BURN RATE
        LESS THAN THE ABSOLUTE VALUE OF THE BURN VECTOR
        ELEMENTS) ARE USED TO INCREMENT THE Z VECTOR
        WHICH INDICATES ENGINE DAMAGE. IF ANY ONE OF THE
        BURN ELEMENTS EXCEEDS M, THE CORRESPONDING
        ELEMENT IN Z IS INCREMENTED.

14 THE RESULTANT B IS LIMITED TO THE MAXIMUM BURN RATE. THE
        OPERATION ××B REPLACES THE SIGN ON EACH ELEMENT
        OF B WHICH WAS REMOVED BY THE ABSOLUTE VALUE OR
        MAGNITUDE OPERATION.

15 THE BURN VALUES (B) ARE LIMITED BY THE AVAILABLE FUEL
        AND FUEL PRIORITY GOES TO THE VERTICAL ENGINE
        BEFORE TRANSVERSE, BEFORE HORIZONTAL. NOTE THE
        USE OF THE PLUS SCAN OPERATOR (+\). IF B=1 2 3,
        THEN +\¯3↑2↓B WOULD BE +\0 1 2 OR 0 1 3. IF PLUS
        SCAN IS NOT IMPLEMENTED LINE 15 MUST BE EXPANDED
        TO 3 LINES WITH SEPARATE FUEL TESTS FOR EACH
        ELEMENT OF B.

16 THE NEW FUEL VALUE IS CALCULATED AND THEN TESTED, A
        BRANCH TO L1530 OCCURS IF MORE THAN 9 THOUSANDTHS
        OF A BURN UNIT REMAIN.

17 PRINTS THE OUT OF FUEL WARNING

18 L1530 CONTAINS A POWERFUL TEST USED TO DETERMINE IF ANY
        ENGINE HAS BEEN ABUSED. TWO LOGICAL VECTORS (0=E
        AND 2=Z) ARE CATENATED AND THE RESULT IS TESTED
        VIA OR REDUCTION (∨/). IF ANY OF THE ELEMENTS IN
        THE RESULTANT VECTOR IS SET TO 1, THE BRANCH TO
        L1780 OCCURS (BANG).

19 SHOWS APL ARRAY CAPABILITY AS THE LOGICAL VECTOR Z≠0 IS
        USED TO SELECT ROWS OF A CHARACTER ARRAY WHICH
        CONTAINS A MESSAGE. IF THE MICRO-APL SYSTEM
        DOESN'T SUPPORT ARRAY OPERATION, USE ONE OF THE
        NON-ARRAY ALTERNATIVES SHOWN BELOW.

   [19]    (0≠1↑Z)/'VERTICAL',(ENG←' ENGINE WILL BLOW IN ')
                ,(⍕1↑E),BN←' BURNS'
   [19.1] (0≠1↑1↓Z)/'TRANSVERSE',ENG,(⍕1↑1↓E),BN
   [19.2] (0≠¯1↑Z)/'HORIZONTAL',ENG,(⍕¯1↑E),BN

        IF VECTOR SUBSCRIPTING IS ALLOWED A CONVENIENT
        ALTERNATIVE IS:

   [19]    0 BURN 'VERTICAL'
   [19.1] 1 BURN 'TRANSVERSE'
   [19.2] 2 BURN 'HORIZONTAL'

        WHERE BURN IS THE FUNCTION...
        ∇A BURN MSG
    [1]  Z[A]/MSG,' ENGINE WILL BLOW IN ',(⍕E[A]),' BURNS'∇

20 LIKE LINE 13 USES THE LOGICAL VECTOR RESULT IN AN
        ARITHMETIC OPERATION-TO DECREMENT THE REMAINING
        BURN COUNT.

21 L860 - THE REAL WORK OF THE FLY FUNCTION! THIS STATEMENT
        CALCULATES THE NEW DISTANCE VECTOR

22 FORMS THE NEW VELOCITY VECTOR

23 ZEROS THE BURN VECTOR IF THE FUEL IS EXHAUSTED.

24 BRANCHES BACK TO L640 IF THE LEM IS MORE THAN A
        MILLIMETER ABOVE THE SURFACE.

25 THE ENDGAME FUNCTION PRINTS YOUR REPORT CARD.

FUNCTION NAME: CKSCALE

        THIS FUNCTION DISPLAYS A GRID SHOWING THE LEM POSITION
        (÷SYMBOL) RELATIVE TO THE TARGET LANDING AREA. THE
        GRID APPEARS AS SOON AS THE ALTITUDE IS 1000 METERS OR
        LESS. WARNING MESSAGES APPEAR AT 100 METERS ALTITUDE
        AND AN EXPANDED WARNING APPEARS AT 10 METERS. TRY
        LINES 6 AND 7 IN BASIC! THE HORIZONTAL AND TRANSVERSE
        SCALES OF THE GRID ARE AUTOMATICALLY ADJUSTED AS THE
        LEM GETS CLOSER TO THE TARGET. THE LEM IS SUPERIMPOSED
        ON THE TARGET IMAGE IF YOU FLY IT WITHIN 4 METERS.
        THIS FUNCTION USES VECTOR OPERATIONS ONLY EXCEPT FOR
        LINE 14 WHICH PRINTS THE GRID AS A CHARACTER ARRAY.
        LINE 14 COULD BE REPLACED WITH A LOOP TO PRINT EACH
        ROW OF THE GRID. THE ROTATE OPERATORS (⌽) CAN BE
        ELIMINATED BY CHANGING A COUPLE OF SIGNS IN LINE 12.
        THIS WHOLE ROUTINE IS MORE STRAIGHTFORWARD IF ARRAY
        INDEXING IS ALLOWED. EG - HERE ARE LINES 9-14 IN APLSV
        NOTATION AS USED IN THE IBM 5100:

        [9]  PLOT←11 11⍴0
        [10] PLOT[;5]←6
        [11] PLOT[5;]←5
        [12] PLOT[5-(5⌊⌊|D[1]÷DS1)××D[1];5+(5⌊⌊|D[2]÷DS2)××D[2]]←1
        [13] PLOT[5;5]←PLOT[5;5]+2
        [14] ' ÷□⌹+|-□□'[PLOT]∇

         AN ALTERNATIVE ROUTINE WHICH USES A LOOP AND AVOIDS
         BOTH VECTOR AND ARRAY INDICES IS:

        [9]  L←5-(5⌊⌊|D1÷DS1)××D1←1↑1↓D
        [10] R←5+(5⌊⌊|D2÷DS2)××D2←1↑D
        [11] I←0
        [12]LOOP:→(I=L)/SET
        [13] →(I=5)/R5A
        [14] 11↑¯6↑'-'
        [15] →NXT
        [16]R5A:11⍴(5⍴'|'),'□'
        [17] →NXT
        [18]SET:→(I=5)/R5B
        [19] (R↑SC),'÷',(R+1)↓SC←11↑¯6↑'-'
        [20] →NXT
        [21]R5B:(R↑SC),(1↑((R=5)/'⌹'),'÷'),(R+1)↓SC←11⍴(5⍴'|'),'□'
        [22]NXT:→(10≥I←I+1)/LOOP∇

FUNCTION NAME: PTSTAT

        THIS FUNCTION PRINTS OUT THE CURRENT FLIGHT
        STATISTICS.

FUNCTION NAME: LEM

        THIS FUNCTION IS USED TO INPUT THE INITIAL PARAMETERS
        TO THE WORKSPACE. NOTE THAT MORE VARIABLES ARE
        REQUESTED IN THIS VERSION OF LEM THAN IN THE BASIC
        VERSION. THE ADDED VARIABLES LET YOU MAKE THE GAME
        MORE CHALLENGING BY SPECIFYING THE MINIMUM BURN TIME
        AND ALSO IMPOSING CORIOLIS FORCES ASSOCIATED WITH THE
        ROTATIONOF THE PLANET IN ADDITION TO THE GRAVITY
        FORCE.

FUNCTION NAME: SETVALUE

        THIS FUNCTION IS USED TO INPUT NEW TIME AND BURN
        VALUES TO THE FLY PROGRAM. SETVALUE USES THE SET
        FUNCTION TO READ AND CHECK EACH VALUE. SET IN TURN
        USES THE FIX FUNCTION TO TRANSLATE THE CHARACTER INPUT
        INTO A VALID NUMERIC VALUE. IF NO VALUE IS ENTERED IN
        RESPONSE TO THE REQUEST FOR INPUT, THE PREVIOUSLY
        ENTERED VALUE WILL BE USED. SET ALSO SCANS FOR A "→"
        IN THE INPUT CHARACTER STRING. THIS CHARACTER IS USED
        TO SHUT DOWN THE ENGINE FOR THE DURATION OF THE
        FLIGHT. THE APPROPRIATE BURN VALUE WILL BE SET TO ZERO
        AND NO FURTHER INPUT VALUES WILL BE REQUESTED FOR THIS
        ENGINE.

FUNCTION NAME: ENDGAME

        THIS FUNCTION PRINTS THE GOOD/BAD NEWS ONCE THE LEM
        HAS REACHED THE SURFACE. NOTE THAT A COMMON MESSAGE IS
        USED FOR ALL CRASH REPORTS WITH LOGICAL OPERATIONS
        USED TO CONTROL INCLUSION OF VARIOUS DISASTERS.
```

Listing 3: A sample run of the Lunar Lander program. This is the classic version of the game in which the operator attempts to become a senior pilot by optimally landing his craft within the constraints of the program. As part of its printout, the program provides a set of two coordinate axes showing the position of the operator's ship at discrete time intervals.

```
      FLY
AT TIME T = 0 SEC. STATS ARE AS FOLLOWS:
          VERTICAL              TRANVERSE              HORIZONTAL
DISTANCE  3000.00               2300.00                2000.00 METERS
VELOCITY  ¯100.00               ¯580.00                   5.00MTRS/SEC.
FUEL...   1500.00 BURN UNITS
COMPUTED TIME TO IMPACT AT PRESENT RATE:...  26.49 SECONDS.
ENTER TIME INTERVAL-  10
     VERTICAL (Z) BURN?  ¯1
     TRANSVERSE (Y) BURN? 61
     HORIZONTAL (X) BURN? ¯11

AT TIME T = 10 SEC. STATS ARE AS FOLLOWS:
          VERTICAL              TRANVERSE              HORIZONTAL
DISTANCE  1900.00               ¯450.00                1500.00 METERS
VELOCITY  ¯120.00                 30.00                ¯105.00MTRS/SEC.
FUEL...   770.00 BURN UNITS
COMPUTED TIME TO IMPACT AT PRESENT RATE...  14.91 SECONDS.
ENTER TIME INTERVAL-  14.5
     VERTICAL (Z) BURN?  1.2
     TRANSVERSE (Y) BURN? 0
     HORIZONTAL (X) BURN? 1

SCALE:X=20,Y=4
     -
     -
     -
     -
     -
|||||□|||||
     -
     -
     -   ÷
     -
```

Listing 3, continued:

```
AT TIME T = 24.5 SEC. STATS ARE AS FOLLOWS:
          VERTICAL           TRANVERSE           HORIZONTAL
DISTANCE   181.03            ¯15.00               82.63 METERS
VELOCITY  ¯117.10             30.00              ¯90.50MTRS/SEC.
FUEL...  738.10 BURN UNITS
COMPUTED TIME TO IMPACT AT PRESENT RATE...  1.54 SECONDS.
ENTER TIME INTERVAL-  1
     VERTICAL (Z) BURN?  0
     TRANSVERSE (Y) BURN? ¯30
     HORIZONTAL (X) BURN? 70

******PREPARE FOR IMPACT
SCALE:X=20,Y=0
     -
     -
     -
     -
     -
||||||□÷||||
     -
     -
     -
     -

AT TIME T = 25.5 SEC. STATS ARE AS FOLLOWS:
          VERTICAL           TRANVERSE           HORIZONTAL
DISTANCE    63.43               .00               27.13 METERS
VELOCITY  ¯118.10               .00              ¯20.50MTRS/SEC.
FUEL...  638.10 BURN UNITS
COMPUTED TIME TO IMPACT AT PRESENT RATE...  .54 SECONDS.
ENTER TIME INTERVAL-  1
     VERTICAL (Z) BURN?  117
     TRANSVERSE (Y) BURN? →
     HORIZONTAL (X) BURN? 3

******PREPARE FOR IMPACT
**********FASTEN RETAINING STRAPS
SCALE:X=4,Y=0
     -
     -
     -
     -
||||||□|÷|||
     -
     -
     -
     -

AT TIME T = 26.5 SEC. STATS ARE AS FOLLOWS:
          VERTICAL           TRANVERSE           HORIZONTAL
DISTANCE     3.33               .00                8.13 METERS
VELOCITY    ¯2.10               .00              ¯17.50MTRS/SEC.
FUEL...  518.10 BURN UNITS
COMPUTED TIME TO IMPACT AT PRESENT RATE...  1.23 SECONDS.
ENTER TIME INTERVAL-  1
     VERTICAL (Z) BURN?  1
     HORIZONTAL (X) BURN? 17.5

******PREPARE FOR IMPACT
**********FASTEN RETAINING STRAPS
SCALE:X=4,Y=0
     -
     -
     -
     -
     -
|||||⊞|||||
     -
     -
     -
     -

AT TIME T = 27.5 SEC. STATS ARE AS FOLLOWS:
          VERTICAL           TRANVERSE           HORIZONTAL
DISTANCE     1.23               .00                ¯.63 METERS
VELOCITY    ¯2.10               .00                 .00MTRS/SEC.
FUEL...  499.60 BURN UNITS
COMPUTED TIME TO IMPACT AT PRESENT RATE...  .52 SECONDS.
ENTER TIME INTERVAL-  1
     VERTICAL (Z) BURN?  1
     HORIZONTAL (X) BURN? →

******PREPARE FOR IMPACT
**********FASTEN RETAINING STRAPS
SCALE:X=4,Y=0
     -
     -
     -
     -
     -
|||||⊞|||||
     -
     -
     -
     -

AT TIME T = 28.5 SEC. STATS ARE AS FOLLOWS:
          VERTICAL           TRANVERSE           HORIZONTAL
DISTANCE      .00               .00                ¯.63 METERS
VELOCITY    ¯2.10               .00                 .00MTRS/SEC.
FUEL...  498.60 BURN UNITS
COMPUTED TIME TO IMPACT AT PRESENT RATE...  .00 SECONDS.
******THE HAL-LUJAH HAS LANDED!******

YOU QUALIFY FOR SENIOR PILOT WITH THAT ONE!
```

■

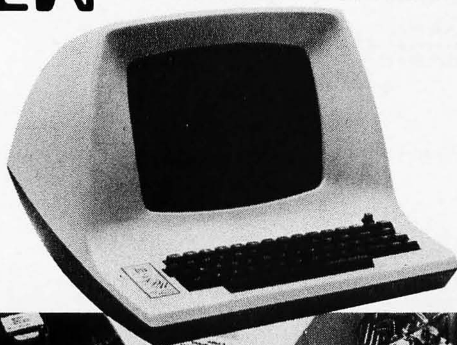
SYNCHRO-SOUND Enterprises

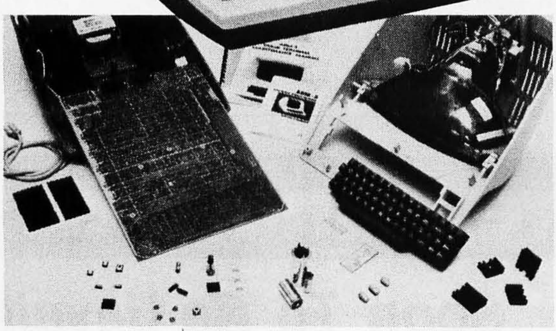
Your Small Computer Department Store

One-stop shopping for Hardware and Software

Compare our PRICES, PRODUCTS, QUALITY, DELIVERY, SERVICE . . . we carry everything you need in small computers, terminals, printers, monitors, keyboards, accessories. We'll meet or beat competitive prices on everything we sell!

NEW LEAR SIEGLER ADM-3A TERMINAL

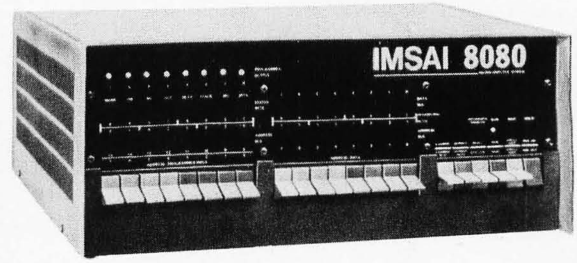




- **Full addressable cursor.**
- **Character Generation— 5 x 7 dot matrix.**
- **Display Format— Standard: 1920 characters, displayed in 24 lines of 80 characters per line.**
- **Character Set— Standard: 64 ASCII characters, displayed as upper case, plus punctuation and control.**
- **Communications Rates— 75 to 19,200 baud (switch selectable).**
- **Computer Interfaces— EIA standard RS232C and 20mA current-loop (switch selectable).**
- **Data Entry— New data enters on bottom line of screen; line feed causes upward scrolling of entire display with top-of-page overflow. Automatic new line switch selectable, end-of-line audible tone.**

ADM-3A Kit	**795.00**
ADM-3A Assembled	**895.00**
Lower Case Option	**89.00**

IMSAI 8080 MICROCOMPUTER



Powerful • Easy to use • Low cost

With 22 Slot Mother Board	**$619.95**
With Z-80 CPU	**849.95**

OKIDATA Model 110 Line Printer



• 110 CPS dot matrix

Friction Feed	**$1099.00**
Tractor Feed	**1229.00**
RS 232C Serial Interface	**250.00**

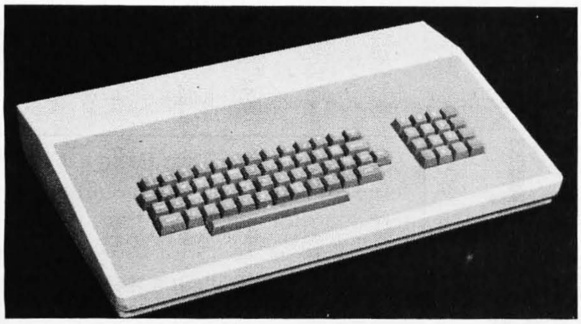
SPECIAL BUYS

HAZELTINE Modular One Video Terminal	**$1639.00**
SORAC IQ120 Video Terminal Kit	**959.00**
DEC LA180 Printer	**2769.00**
COMPUCOLOR 8001 Color Graphics Computer	**2595.00**
ICOM Microfloppy Disc & Controller	**989.00**
NORTH STAR Microfloppy Disc & Controller	**599.00**
OKIDATA Model 22 Printer	**2095.00**
OMNITEC 401C 300 Baud Modem	**300.00**
2708 EPROMS	**29.95**
OAE Papertape Reader	**69.95**

SOL 20 In Stock For Immediate Delivery

NEW

COMMERCIAL QUALITY KEYBOARD

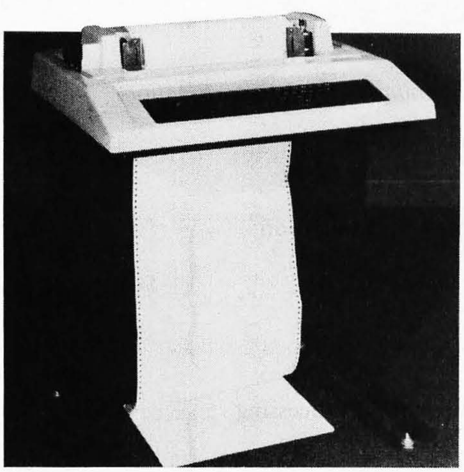


The Model SS-1 Communications Terminal is a non-contact capacitive keyboard with a guaranteed life of over 100,000,000 operations.

- **MOS/LSI Encoder with high reliability low power consumption.**
- **Key Roll Over which eliminates operator error and increases thruput.**
- **Hysteresis for tease proof operation.**
- **Solid State performance at mechanical switch prices.**
- **Tactile Feedback at the operate point.**
- **ASR-33 Array with four mode encoding.**

Keyboard Kit........ **$99.95**
Enclosure (with numeric cutout)............ **49.95**
Numeric Pad......... **34.95**
Complete Kit (with pad) **179.95**
Assembled Unit (with pad)......... **229.95**

DECwriter II



- **132 column printing • 30 CPS**
- **Full keyboard • Tractor Feed**

$1769.95

SOFTWARE

Attention ALTAIR DISK Owners

Now available—an Accounts Receivable System for the small user. This system is expressly designed to be run on an 8080 based microcomputer using an Altair floppy disk with Altair Basic, Video Terminal and Printer. Many features normally found only on large scale computers are included. Provided are such functions as: Adding New Customer Accounts, Deleting Dormant Accounts. Transaction Processing, and Report Generation. The Monthly Statements include both current and aged balances. Other reports generated are the Aged-Accounts Receivable and Delinquency Notices.

This easily usable system requires very little operator training. All communication is in an interactive mode with the program constantly prompting the appropriate responses. Operator errors are easily corrected and accidental data base deletions are prevented by requiring additional confirmation.

This Accounts Receivable System, although supplied ready for use, may readily be modified to include special installation-dependent functions. All major programs are written in high level language making program additions and alterations a simple task. This program module is designed to serve as the basis for a complete accounting package. Provisions are included for linking Billing Inventory Control and General Ledger modules which will be available in the near future.

We carry a full line of the following: TDL, OAE, Processor Tech., Hayden, Tarbell, IMSAI, Cromemco, Compucolor, Icom, Lear Siegler, Okidata, DEC, Javelin, Teletype, North Star, Heuristics, Peripheral Vision, TRW. Same day delivery and shipping on most items. Full, modern repair facilities on premises for complete servicing of everything we sell.

SYNCHRO-SOUND Enterprises
The Computer People
193-25 Jamaica Avenue, Jamaica,
New York 11423 • 212/468-7067

Hours 9-4 daily and Saturday

Visit our new showroom . . . working units on display

Dept. B BankAmericard • Master Charge

☐ Please send me further information on: (list items)

____

______________________________ Price List ☐

Name ______________________________

Address ______________________________

City ____________ State ____________ Zip ________

Circle 118 on inquiry card.

An APL Interpreter for Microcomputers: Part 1

Michael Wimble
6026 Underwood Av SW
Cedar Rapids IA 52404

With this article we begin a three part series by Mike Wimble about an APL interpreter for small systems. The articles consist of a series of flowcharts from which the enterprising reader can construct his or her own code. Considering the complexities of APL, it is interesting to note that Mike's own program for this interpreter (written in PL/I) is less than 1600 statements long!

The format of this article will be slightly different from most of the other articles that have appeared in BYTE. Because this series of articles is predominantly pictorial, we have integrated the main text into the figure captions. As a result, some of the captions discuss topics related to other figures. In these cases we have added cross references for clarity's sake. We hope this format will eliminate unnecessary page turning.

Table of Contents

The following table of contents describes the material covered in this month's article as well as the two upcoming articles (to be published in the September and October issues of BYTE).

The Great APL Interpreter Contest

As an incentive to those experimenters who would like to try writing their own APL interpreters based on this series of articles, BYTE announces the Great APL Interpreter Contest. We will award prizes for APL interpreters (suitable for publication with royalties to authors) based on Mike's flowcharts (or independent of them if you prefer).

Contestants are free to write their interpreters for any microprocessor they choose. Entries will, however, be judged on their suitability for use on small systems with a minimum of 16 K bytes of memory, as well as on programming elegance and efficient use of space. All of these factors should therefore be kept in mind.

Entries should be addressed to BYTE, attn: The Great APL Interpreter Contest, 70 Main St, Peterborough NH 03458, and must be postmarked no later than midnight, February 28 1978. Entries must be in the form of a publication quality manuscript which describes the implementation of the interpreter and which includes a listing of source code and object code. Contestants should also submit machine readable source and object code in the form of paper tape or cassette.

The winners (if any) will receive $1000 plus normal author payments, should the entry be chosen for publication in book form or as an article in BYTE. We reserve the right to choose more than one winner under the same terms.

Judging will be done by the editors of this magazine. Those seriously interested in entering this contest should call Carl Helmers or Chris Morgan at BYTE, (603)924-7217. May the midnight oil burn prosperously for all.

Introduction

Under the sponsorship of the National Science Foundation, Bernard J Robinet produced an NTIS report entitled, "Architectural Design of a Directly-Executed APL Processor". Although at a detail level the paper is fraught with logical and typographical errors, it presents an excellent and complete plan for implementing an APL interpreter via software, firmware or hardware. This article is a description of just such an interpeter. The program was originally written in the PL/I language but is presently being enhanced and translated for execution on a TMS 9900 microprocessor. No attempt will be made herein to describe APL, and the information is aimed at the experienced programmer who can use flowcharts and diagrams as a starting point. Emphasis is placed on architecture, rather than on actual code so that, while the more experienced programmer should have enough information to program his or her own interpreter, even the neophyte can still learn many of the techniques of interpreter design.

The APL interpreter is organized as two separate subprograms which are executed sequentially. The first subprogram is the scanner, the second is the intepreter. The scanner accepts all program input, translating the program into a series of syllables and pointers, until the end of an immediate mode statement is found. The interpreter gains control to execute an immediate mode statement, using the information built by the scanner, and returns control to the scanner upon completion. We begin our discussion by looking at some of the tables common to both subprograms. After this discussion one should have a general understanding of the working of the interpreter and we can begin a detailed description of the actual program.

Table 1: A sample APL statement, plus a breakdown of the statement into "tokens." This program inputs a list of numbers and prints their average. The token is the basic semantic element of a program. Just as a program consists of one or more statements, a statement consists of one or more tokens. (In the literature, the term token is sometimes equated with "symbol" or "atom." This can lead to some confusion because the term "atom" has a somewhat different meaning when used by LISP programmers.) Even though tokens can be further broken down into elements called characters, the interpreter spends very little time dealing with individual characters compared with the time spent in manipulating tokens.

APL Statement	Characters	Token
⎕←(+/XX)÷ρ,XX←⎕	⎕	quad
	←	left arrow
	(	left parenthesis
	+	plus operator
	/	reduction operator
	XX	variable name
	)	right parenthesis
	÷	divide operator
	ρ	rho operator
	,	ravel operator
	XX	variable name
	←	left arrow
	⎕	quad

Table 2: Set of noun-adjective combinations (syllables) used by the interpreter. One of the main functions of the scanner is to extract the tokens from each statement, left to right, and to store them in a table as a noun-adjective combination. Surprisingly few nouns (only nine) are needed to construct a powerful interpreter. The table of noun-adjective pairs built by the scanner is labelled SP, and each entry consists of two elements: C is the noun and P is the associated adjective. Three of the adjectives are labelled "variable number." This number is actually a pointer to the symbol table VTAB which is indexed by VAR. Each entry in VTAB has three elements: V1, V2 and V3. V1 is the character representation of the symbol. V2 is a pointer to another table, FTAB (see table 5), but V2 is used to distinguish between global variables and formal parameters of the same name. V3 is also a pointer to another table, D (see tables 4a and 4b), and is set by the statement interpreter as memory is assigned to hold a value for the variable. V3, then, is either the address, or a pointer to the address, of the value of the variable.

NOUN NUMBER	NOUN	ASSOCIATED ADJECTIVE
1	FORMAL RESULT	VARIABLE NUMBER
2	FORMAL ARGUMENT	VARIABLE NUMBER
3	GLOBAL VARIABLE	VARIABLE NUMBER
4	MONADIC OPERATOR	OPERATOR NUMBER
5	REDUCTION OPERATOR	OPERATOR NUMBER
6	DYADIC OPERATOR	OPERATOR NUMBER
7	SPECIAL OPERATOR	CHARACTER
8	CONSTANT	ADDRESS OF CONSTANT
9	STATEMENT END	STATEMENT NUMBER

Table 3: Representation of the tokens of the APL statement ⎕←(+/XX)÷ρ,XX←⎕ in table SP (see table 1). After a program has been stored left to right in table SP by the scanner, the interpreter will execute the statements from right to left. Obviously more information is needed for statement execution than is contained in this table.

TABLE SP INDEXED BY I

ENTRY	C	P	TOKEN	
1	7	11	QUAD	⎕
2	7	3	LEFT ARROW	←
3	7	6	LEFT PARENTHESIS	(
4	5	1	REDUCTION BY +	+/
5	3	1	GLOBAL VARIABLE XX	XX
6	7	7	RIGHT PARENTHESIS	)
7	6	4	DYADIC OPERATOR ÷	÷
8	4	7	MONADIC OPERATOR ρ	ρ
9	4	8	MONADIC OPERATOR ,	,
10	3	1	GLOBAL VARIABLE XX	XX
11	7	3	LEFT ARROW	←
12	7	11	QUAD	⎕

Table 4a: Format of entry in table D for a vector or a matrix. When a constant is encountered in a statement, the actual value is stored in this table, and the appropriate adjective in table SP points to the address of the constant in this table. Constants, values assigned to variables, and intermediate results of computations are stored in this table. Each entry in the table is a variable length string consisting of several elements.

TABLE D INDEXED BY DA

ELEMENT	CONTENTS
1	TOTAL LENGTH OF ENTRY
2	NUMBER OF DIMENSIONS FOR ENTRY (RANK OF ENTRY OR ρρENTRY)
3	LENGTH OF DIMENSION 1 (SIZE OF FIRST DIMENSION OR ρENTRY[1])
4	LENGTH OF DIMENSION 2 (SIZE OF SECOND DIMENSION OR ρENTRY[2])
N+2	LENGTH OF DIMENSION N (SIZE OF LAST DIMENSION OF ρENTRY[ρρENTRY])
N+3 . . N+2+×/ρENTRY	VALUE OF ELEMENTS STORED IN ROW MAJOR ORDER

Table 4b: Format of entry in table D for a scalar.

TABLE D INDEXED BY DA

ELEMENT	CONTENTS	
1	CONSTANT 3	(LENGTH OF ENTRY)
2	CONSTANT 0	(NUMBER OF DIMENSIONS FOR ENTRY)
3	VALUE OF SCALAR	

Table 5: Format of an entry in the FTAB table. This table is used to store function headers when they are encountered. Each entry has seven elements, as shown. The remaining table that deals with function definitions is table VFUNC (table 6). There are also several tables of constants, but they are used primarily to simplify the scanner, and are discussed as necessary in the section entitled Statement Scanner later in the article.

TABLE FTAB INDEXED BY F

ELEMENT	USE
F1	CHARACTER REPRESENTATION OF FUNCTION NAME
F2	"ARITY" OF FUNCTION. EQUALS 0 FOR NILADIC, 1 FOR MONADIC, AND 2 FOR DYADIC.
F3	"EXPLICIT RESULT" FLAG. BINARY 0 FOR NO EXPLICIT RESULT, BINARY 1 FOR EXPLICIT RESULT.
F4	CHARACTER REPRESENTATION OF VARIABLE NAME OF EXPLICIT RESULT.
F5	CHARACTER REPRESENTATION OF VARIABLE NAME OF LEFT ARGUMENT FOR DYADIC FUNCTION.
F6	CHARACTER REPRESENTATION OF VARIABLE NAME OF RIGHT ARGUMENT FOR DYADIC FUNCTION, OR ARGUMENT FOR MONADIC FUNCTION.
F7	POINTER INTO TABLE VFUNC.

Note: "ARITY" is a mathematical term denoting the number of arguments in a function.

Table 6: Organization of table VFUNC. VFUNC, indexed by VAL, holds the address pointers to the individual statements in a function. The sample program and comments in listing 1 should help to clarify the use of VFUNC. Two more tables are discussed in parts 2 and 3 of this series of articles.

ENTRY	VALUE
1	NUMBER OF STATEMENTS IN FUNCTION 1.
2	POSITION OF RIGHTMOST SYLLABLE IN TABLE 'SP' OF STATEMENT 1 IN FUNCTION 1.
3	POSITION OF RIGHTMOST SYLLABLE IN TABLE 'SP' OF STATEMENT 2 IN FUNCTION 1.
.	
.	
.	
N+1	POSITION OF RIGHTMOST SYLLABLE IN TABLE 'SP' OF STATEMENT N IN FUNCTION 1.
N+2	NUMBER OF STATEMENTS IN FUNCTION 2.
N+3	POSITION OF RIGHTMOST SYLLABLE IN TABLE 'SP' OF STATEMENT 1 IN FUNCTION 2.
.	
.	
.	

TABLE 7A

MOPTAB

	M1	*M2*
1	+	2
2	-	3
3	×	4
4	÷	5
5	⋆	6
6	ι	21
7	ρ	22
8	,	23

TABLE 7B

DOPTAB

	D1	*D2*
1	+	52
2	-	53
3	×	54
4	÷	55
5	⋆	56
6	ι	87
7	ρ	88
8	,	89
9	=	71
10	≠	72
11	<	73
12	≤	74
13	≥	75
14	>	76
15	∧	77
16	∨	78

TABLE 7C

REDTAB

	R1	*R2*
1	+/	2
2	-/	3
3	×/	4
4	÷/	5
5	⋆/	6
6	=/	21
7	≠/	22
8	</	23
9	≤/	24
10	≥/	25
11	>/	26
12	∧/	27
13	∨/	28
14	⌈/	29
15	⌊/	30
16	⍟/	31

TABLE 7D

CHARTAB

	C1
1	:
2	→
3	←
4	⍀
5	.
6	(
7	)
8	[
9	]
10	:
11	⎕
12	*SPACE*

Tables 7a, 7b, 7c and 7d: These tables complete the set of APL tokens used by this APL interpreter. The tables are scanned by subroutine OTHERS when a token is encountered which has not already been recognized by subroutines IDEN or NOMBRE.

Listing 1: Results of a hypothetical program containing a user-defined function. This exercise demonstrates the interrelationships of tables 1 through 7. The program consists of a definition for a function AVG with a monadic argument Y and an explicit result X. Following the function definition is an immediate mode call of the function.

The scanner begins by inserting a *statement end* noun type in table SP, position 1, and proceeds to scan the first statement. When the function header is recognized, an entry is made in table FTAB as shown and scanning continues with the second statement. Again a statement end noun is inserted into table SP, position 2, but this time it is known that this should be statement number 1 of the function, so the adjective has a value of 1. Positions 3 thru 14 are the noun-adjective description of the second program statement and are similar to table 3, the differences being due mainly to the fact that the variables are no longer global but rather are formal arguments and formal results. Program statement 3 produces no entry in table SP but merely signals the end of a function definition. Table SP, positions 15 thru 19, contains the noun-adjective description of the immediate statement. Note how position 19 indicates a constant whose value is stored in table D beginning at position 1. Also note how the element vector is stored in D and relate this to the description in table 4a.

For the one entry in FTAB, element F7 points to position 1 in table VFUNC (see table 6). Here, VFUNC shows that there is only one statement in the function and that its rightmost syllable (noun-adjective pair) is at position 14 in table SP. Positions 3, 7, 12 and 18 in table SP refer to variables, and the associated adjectives point to entries in table VTAB. For variables X

Results of Scanning a Hypothetical Program Containing a User Defined Function

PROGRAM:

```
       ∇ X ← AVG Y
[1]      X ← (+/Y) ÷ ρ , Y ← ⎕
[2]      ∇
         ⎕ ← AVG 37 ¯25
```

Note: Colored arrows indicate pointer destinations

TABLE SP INDEXED BY I

I = 20

	C	*P*	
1	9	0	
2	9	1	
3	1	1	*X*
4	7	←	←
5	7	(	(
6	5	1	+/
7	2	2	*Y*
8	7	)	)
9	6	4	÷
10	4	7	ρ
11	4	8	,
12	2	2	*Y*
13	7	←	←
14	7	⎕	⎕
15	9	0	
16	7	⎕	⎕
17	7	←	←
18	3	3	*AVG*
19	6	1	3 7 ¯2 5

TABLE VTAB INDEXED BY VAR

VAR = 4

		V1	*V2*	*V3*
1		*X*	1	0
2		*Y*	1	0
3		*AVG*	0	0

TABLE FTAB INDEXED BY F

F = 2

	F1	*F2*	*F3*	*F4*	*F5*	*F6*	*F7*
1	*AVG*	1	*TRUE*	*X*		*Y*	1

TABLE VFUNC INDEXED BY VAL

VAL = 3

1	1
2	14

TABLE D INDEXED BY DA

DA = 8

1	7
2	1
3	4
4	3
5	7
6	¯2
7	5

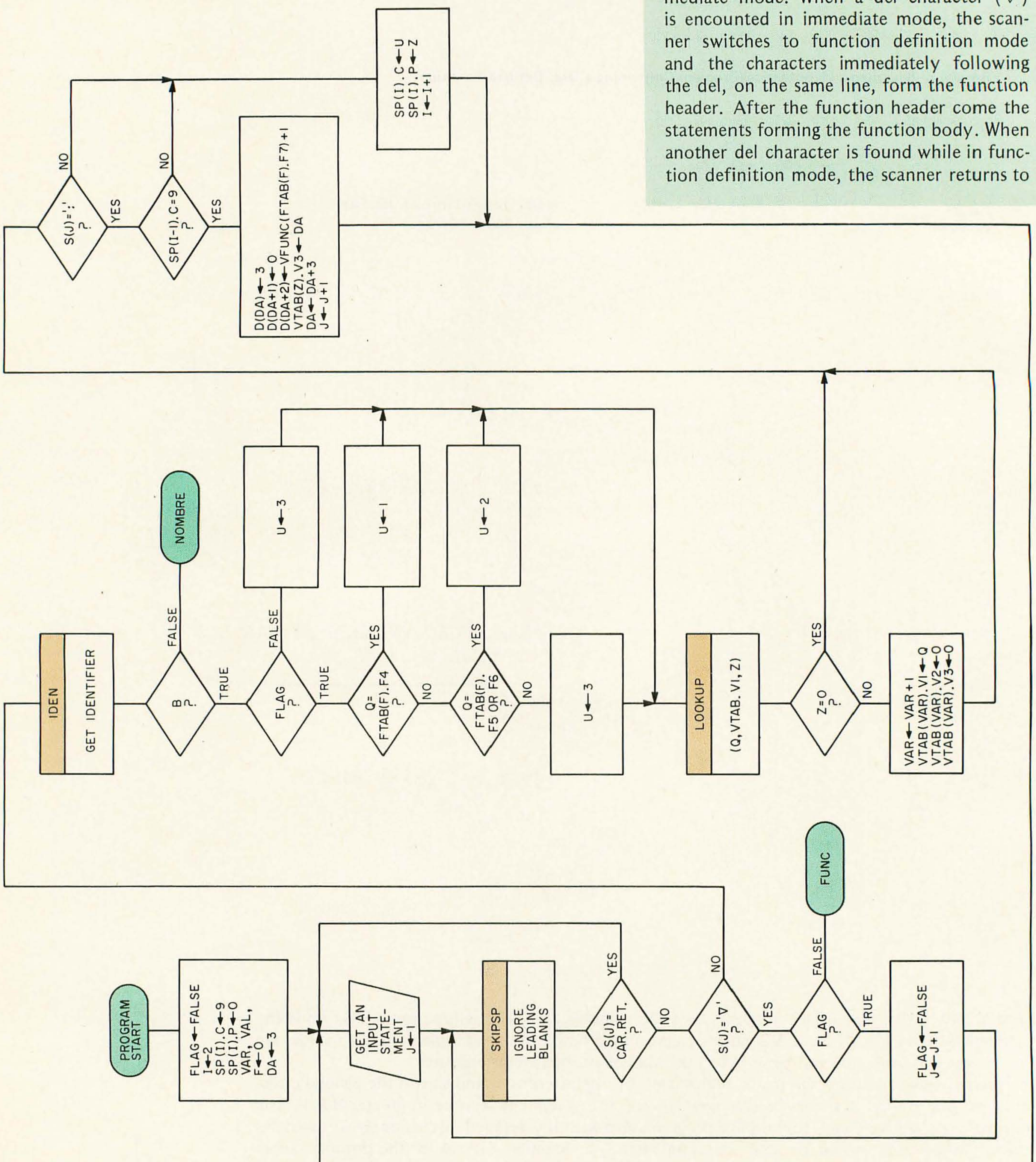
and Y in VTAB, element V2 indicates that the symbols are local to function AVG, while the symbol AVG in VTAB has a V2 value of zero indicating a global variable name. V3 values are all zero at this time, and will be set when the statement interpreter executes.

Having now discussed the major tables used by the interpreter and shown the general direction or goal of the scanner we can now discuss the statement scanner in greater detail. The scanner is a relatively easy function to program, and with the detailed discussion and flowcharts which follow one should be able to implement a scanner for any of the popular small computers.

The Statement Scanner

The main function of this module is to build the various tables required by the statement interpreter. There are two types of statements and two modes of scanning to be handled by this module. The first statement type is the function header which defines the form of a function and which also causes the scanner to enter the function definition mode. The second type of statement is any statement other than a function header. This type of statement can occur when the scanner is in either immediate or function definition mode.

Upon startup, the scanner is in immediate mode. When a del character (▽) is encountered in immediate mode, the scanner switches to function definition mode and the characters immediately following the del, on the same line, form the function header. After the function header come the statements forming the function body. When another del character is found while in function definition mode, the scanner returns to



immediate mode. When a statement is finally scanned in immediate mode, control is transferred to the statement interpreter to execute the statement, using any previously defined functions as necessary. Following execution of the immediate statement, control is again returned to the statement scanner and the process repeats. When a line of input code is read, it is stored in a character vector called S, indexed by J. Each element in S is one character long; the last character for each line must be a carriage return. Thus the expression S(J) refers to the Jth character of the input string. Similarly the expression S(J) = CR is a test for the Jth character being a carriage return (CR). After some initialization, then, block IP loads vector S with the next statement and sets index J to 1.

Subroutine SKIPSP (see figure 2) is next invoked to skip over any leading spaces. If the next character is a carriage return, then the line was all blank and is ignored. Otherwise a check is made for a del character, which causes switching between the two scanning modes as mentioned above. If FLAG is true, the scanner is in function mode and a del will return the scanner to immediate mode. If FLAG is false, the scanner is in immediate mode and a del will put the scanner into function definition mode, and processing will proceed at the box labelled FUNC which scans the function header. If no del character is found, scanning proceeds as normal. The next step is to call subroutine IDEN (figure 3) which is invoked to determine if the next token is an identifier. If not, scanning will resume at block NOMBRE (see figure 4). When an identifier is found, it must be encoded and entered into the appropriate tables. The identifier can be one of four things: a global variable, a formal argument, a formal result, or a label. If FLAG is false (immediate mode), then the identifier can be neither a formal argument nor formal result. If FLAG is true (function definition mode), a quick test will determine the type of identifier: if the identifier is equal to element F4 in table FTAB, the identifier is a formal result of the current function. Similarly, the identifier being equal to elements F5 or F6 in table FTAB means the identifier is a formal argument of the current function.

One must check two things to test if an identifier is a label: if the character immediately after the identifier is a colon (ie: S(J) = ':') and no other semantic information has yet been scanned for this statement (ie: SP(I-1).C=9), then a label has been found. In any case, the identifier must be put into the symbol table, if not already there, and the appropriate noun-adjective combination must be entered in table SP. Finally, if a label was found, the value link (V3) for the symbol table entry is made to point at a scalar in table D. The value of the scalar is equal to the statement numbers where the label is found.

If the next token is not an identifier, scanning continues at the block labelled NOMBRE, as mentioned previously. Figure 4 shows the scanning process for numbers in the source program. Subroutine NUMBER is invoked to extract a number (see figure 5). If the next token is not a number, scanning proceeds at block OTHERS, described next; otherwise the number is stacked in a temporary vector, TVAL. The stacking process continues as long as numbers separated by spaces are found. Finally, the noun-adjective pair for a number is entered into table SP, and the appropriate scalar or vector is created in table D.

OTHERS, shown in figure 6, handles the scanning for tokens not previously recognized. Tables 7a, 7b, 7c and 7d aid in this processing. A call to subroutine LOOKUP appears several times with the accompanying parentheses containing three arguments. Subroutine LOOKUP will scan the table named in the second argument for an occurrence of the first argument: if found, the third argument points to the relative position of the match; otherwise, the third argument is set to zero.

The next step, then, is to check if the next character is in the table of monadic operators. If S(J) is not a monadic operator, a further test is performed to determine if it is a dyadic operator, carriage return, or valid special character. Figure 6 illustrates the handling for each case. If a monadic operator is found, a further test is made to determine if it is part of a reduction character pair. Again, figure 6 shows the testing and handling of each case. The handling of a carriage return will be discussed in the last paragraph of this section.

Returning to the beginning of the discussion of the statement scanner, you will recall that function headers require a special scan. Figures 1, 7a, 7b, 7c, 8 and 9 detail this processing, which is actually quite simple, although at first glance it may appear to be lengthy. The tokens after the del character (del indicates the beginning of a function header) are examined to determine the type of function, its name, its arguments and its result. There may of course be zero, one or two arguments (corresponding to a niladic, monadic and dyadic function respectively),

and one or no explicit result. After these pieces of information have been extracted, they are used to build an entry in the table FTAB, one entry for each function. The symbol table VTAB is updated to reflect the occurrence of any formal parameters.

There comes a point, finally, when a carriage return is found and the scanner is in immediate mode. This occurrence causes the statement scanner to end and the statement interpreter to begin. All the tables described in the first part of this article are now complete, and the variable I points to the first syllable to be interpreted. Next month we will begin to describe the statement interpreter. You should, however, at this point be able to take any APL program and produce a valid set of encoded tables using the flowcharts shown. Do not be intimidated by the number of flowcharts used, since the actual amount of code required for implementation is not too lengthy when using any of the more powerful microprocessors.

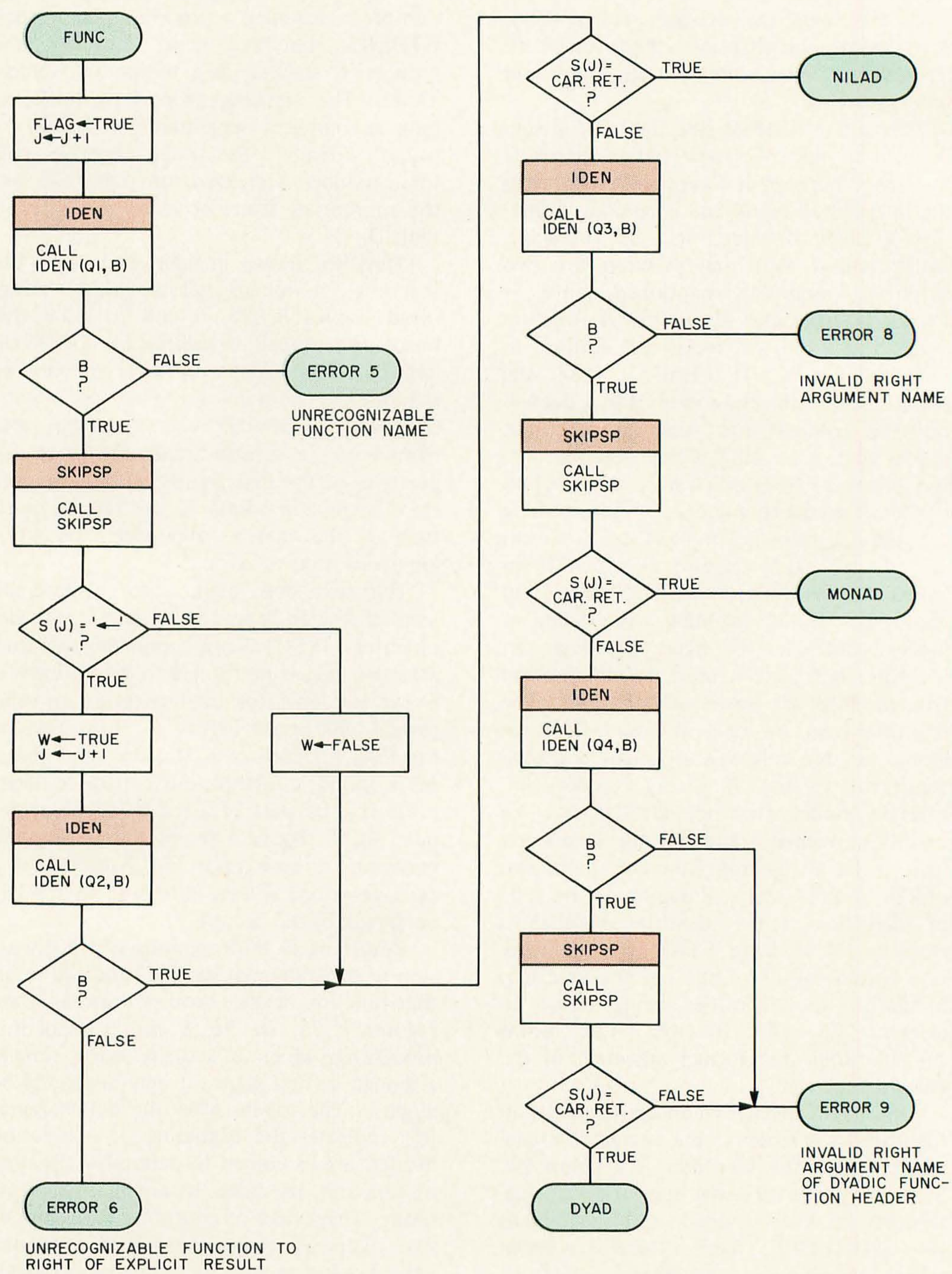
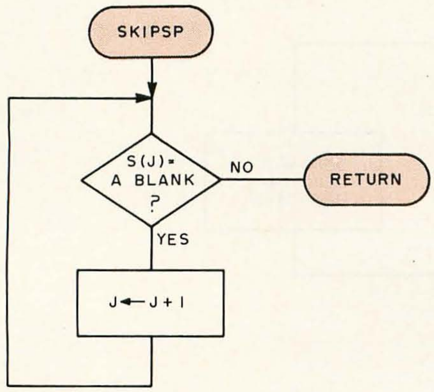


Figure 1: Subroutine FUNC. This routine is called whenever the scanner encounters a del function, which signifies that the following statement contains a function header. The function header contains the syntactic form and the name of a user defined function (see listing 1). The purpose of FUNC is to perform the scanning operation for this special case.

Figure 2: Subroutine SKIPSP. This routine is used to skip over leading blanks.



Call SKIPSP

Result: Global variable J is incremented until S(J) is a nonblank character.

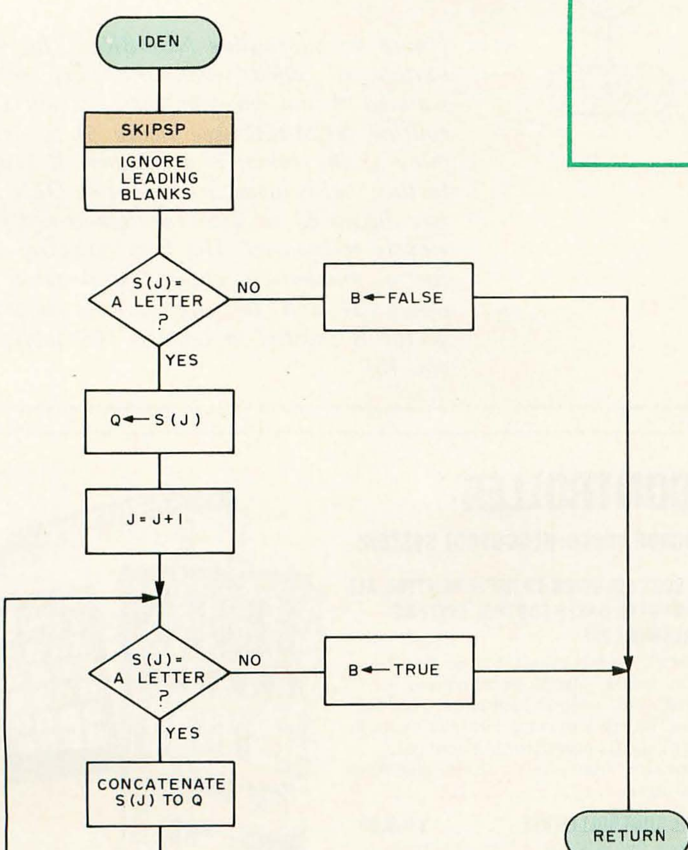


Figure 3: Subroutine IDEN. This subroutine is invoked to determine if the next token is an identifier. If not, scanning resumes at block NOMBRE (see figure 4). If a variable is found as the next token, B will be true and Q will contain the character representation of the variable.

Call IDEN (Q, B)

Result: If a variable is found as the next token, B will be true and Q will contain the character representation of the variable.

If the next token is not a variable, B will be false.

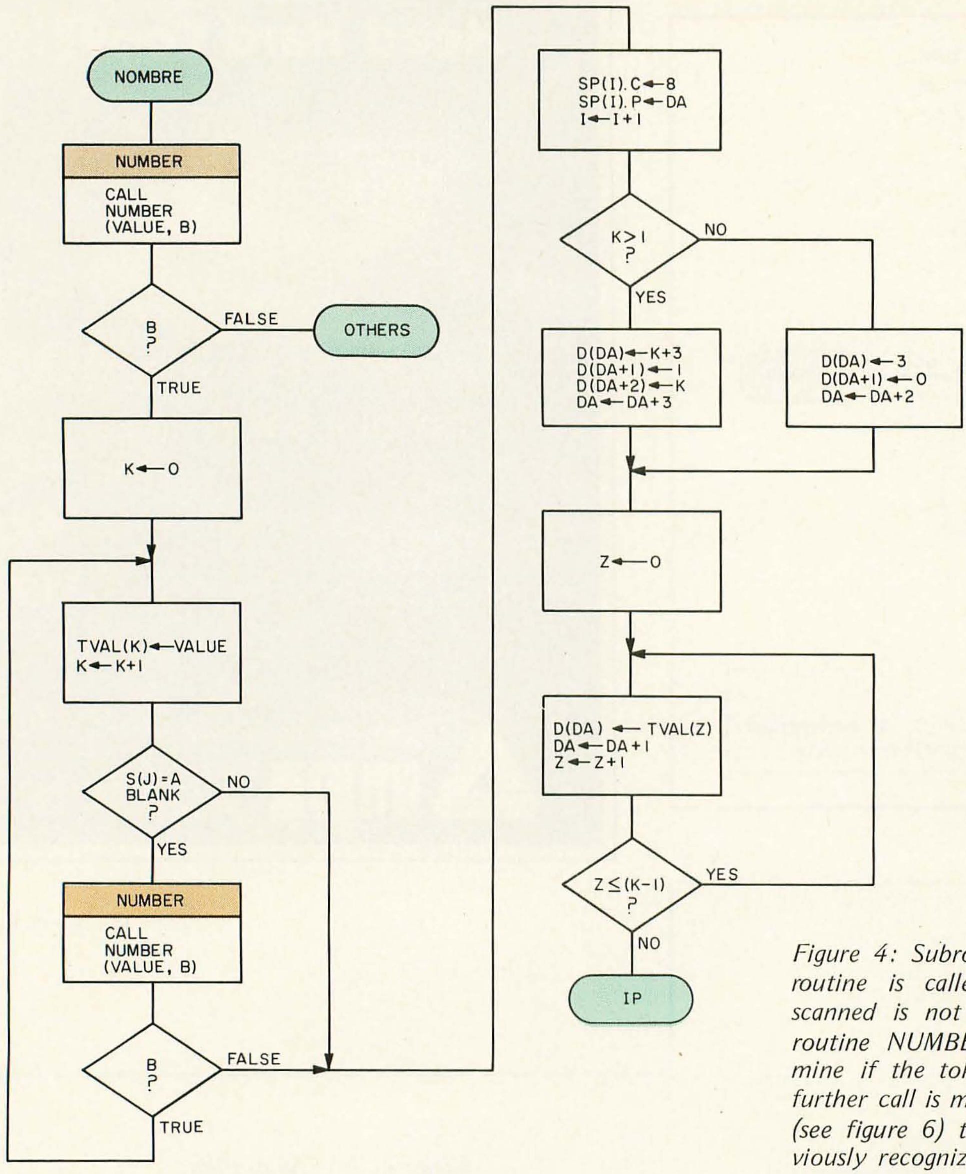
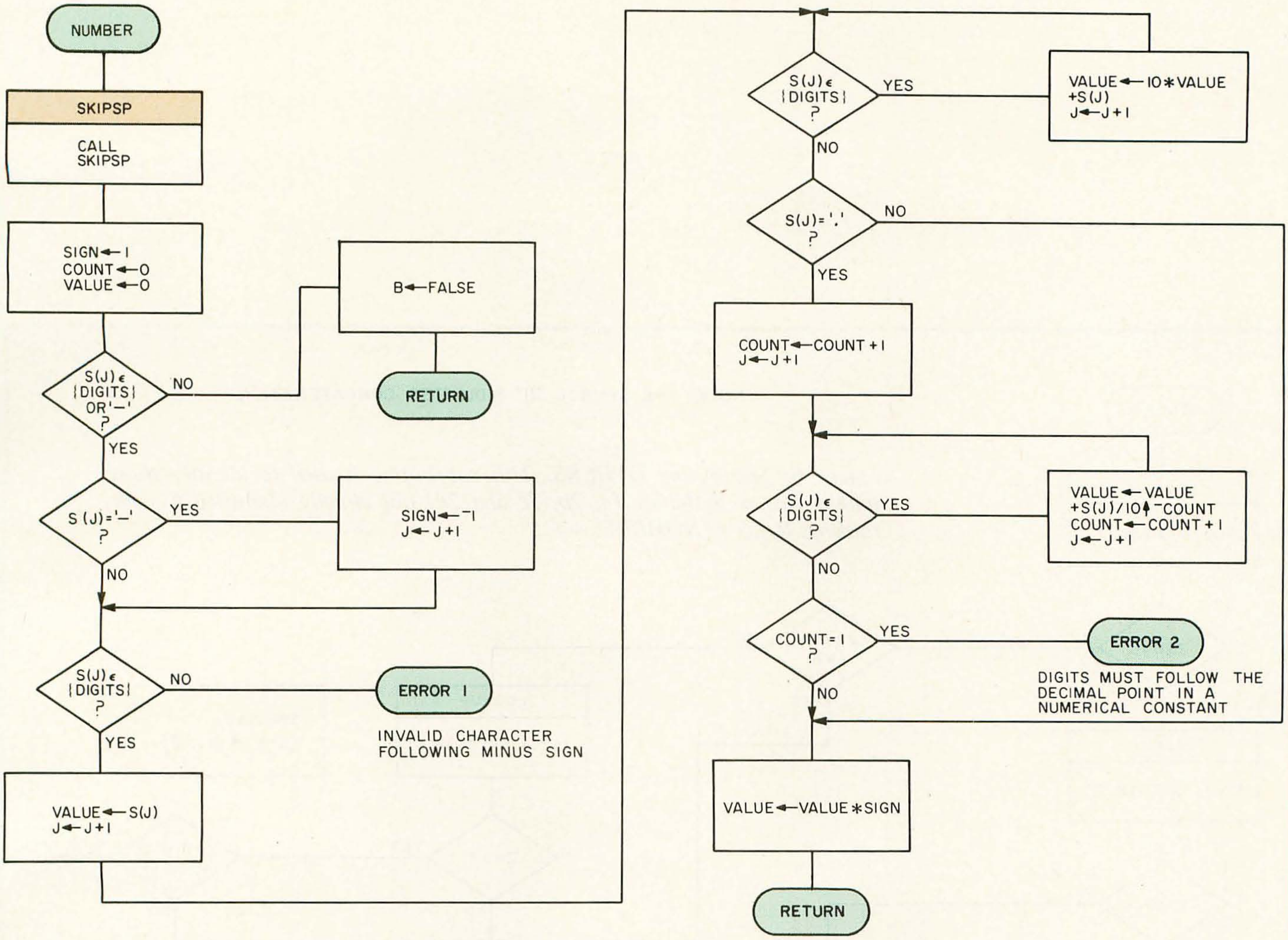


Figure 4: Subroutine NOMBRE. This subroutine is called when the next token scanned is not an identifier. It calls subroutine NUMBER (see figure 5) to determine if the token is a number. If not, a further call is made to subroutine OTHERS (see figure 6) to scan for tokens not previously recognized. The noun-adjective pair for a number is entered into table SP (table 3) and the appropriate scalar or vector is created in table D (see tables 4a and 4b).



Call Number (VALUE, B)

Result: If a numeric constant is found as the next token, B will be true and VALUE will contain the value of the constant.

If the next token is not a numeric constant, B will be false.

Figure 5: Subroutine NUMBER. This subroutine tests the token being scanned to determine whether it is a number or not. If a numerical constant is found, B will be true and VALUE will contain the value of the constant. If the next token is not a number, B will be false.

Circle 121 on inquiry card.

NOTE: THE SYMBOL "||" INDICATES CONCATENATION.

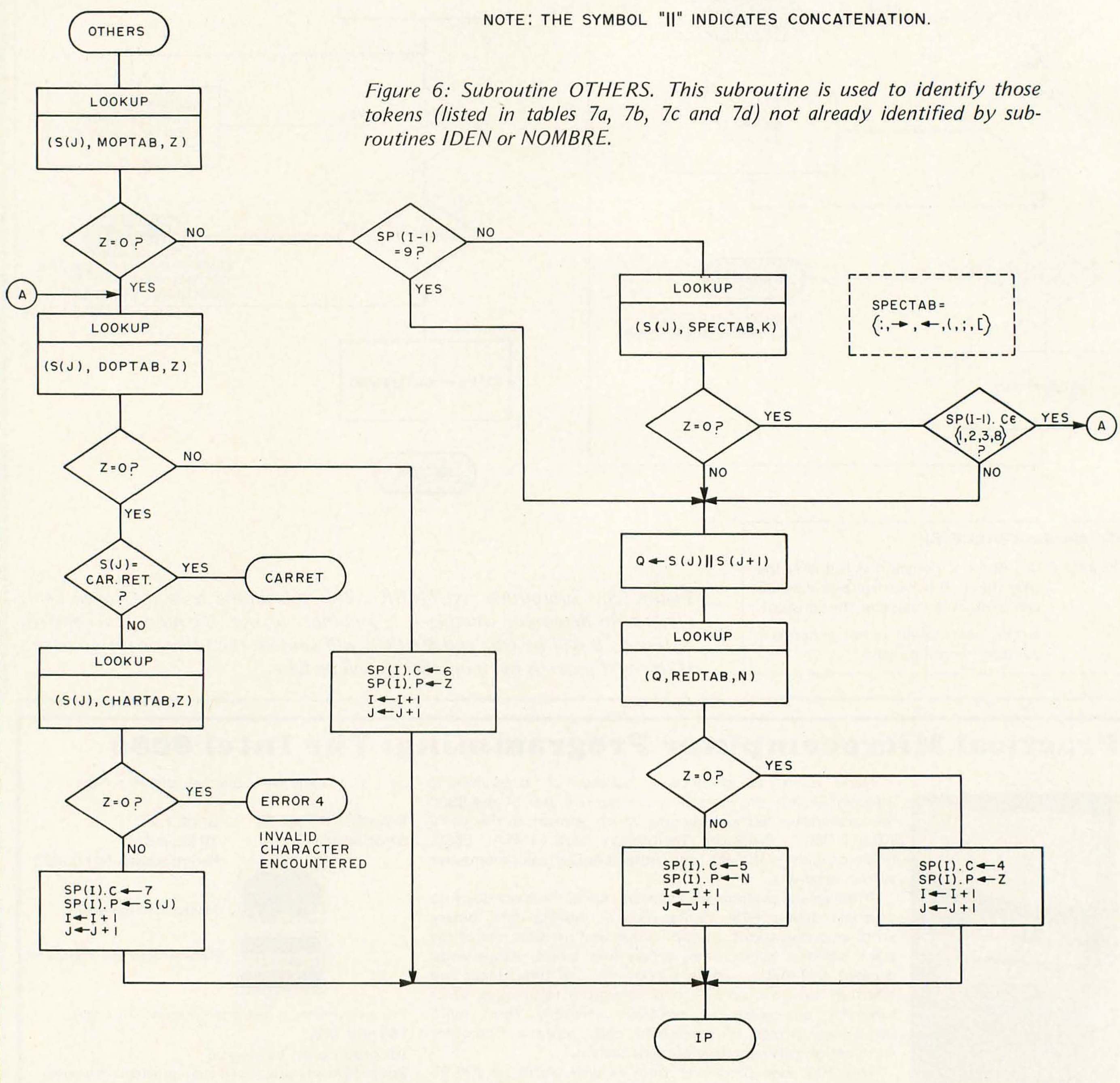


Figure 6: Subroutine OTHERS. This subroutine is used to identify those tokens (listed in tables 7a, 7b, 7c and 7d) not already identified by subroutines IDEN or NOMBRE.

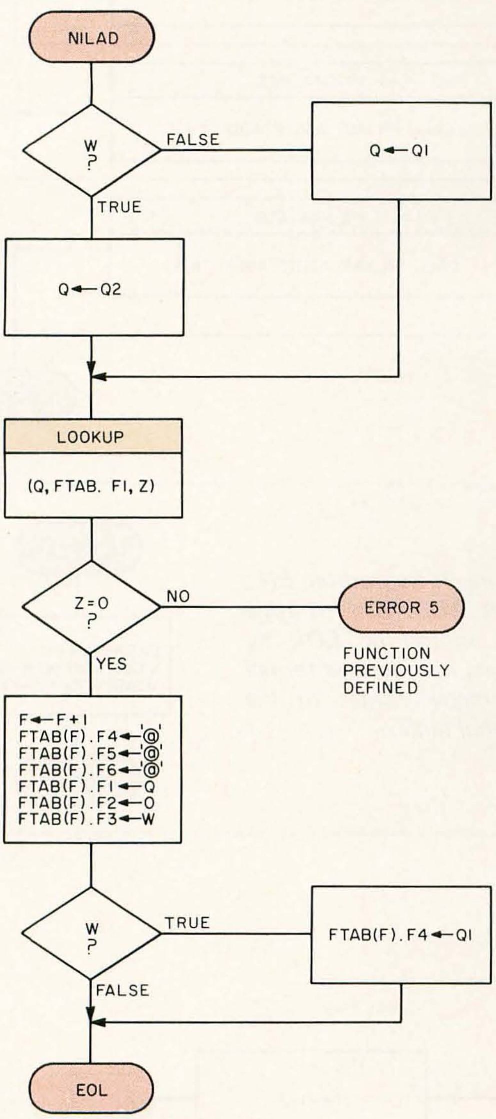


Figure 7a: Subroutine NILAD. This subroutine performs the scanning function for a niladic function header. A niladic function is one which takes no arguments.

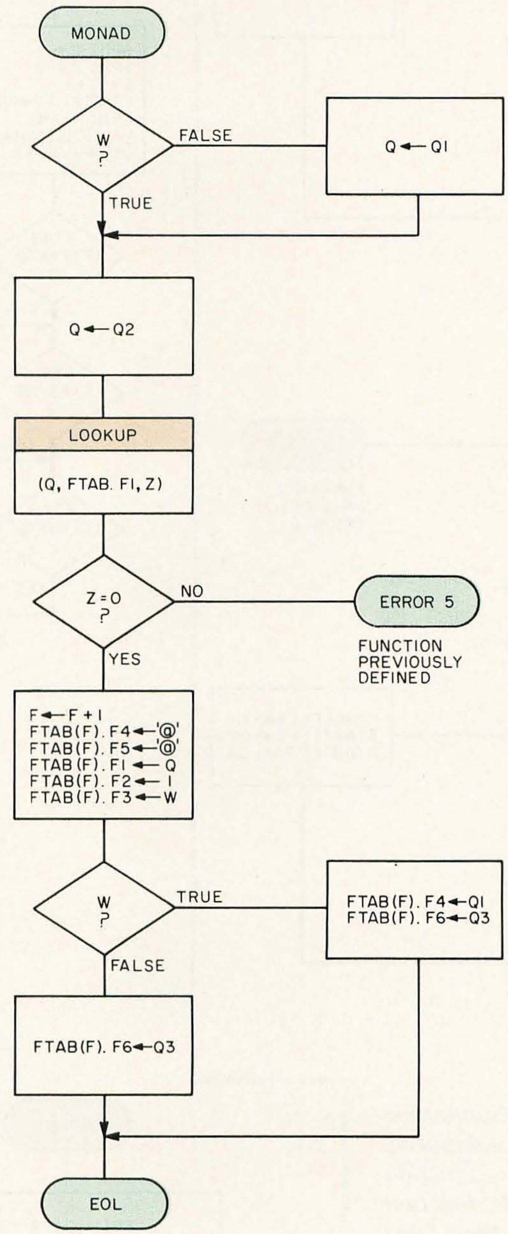


Figure 7b: Subroutine MONAD. This subroutine scans monadic function headers. A monadic function is one which takes one argument.

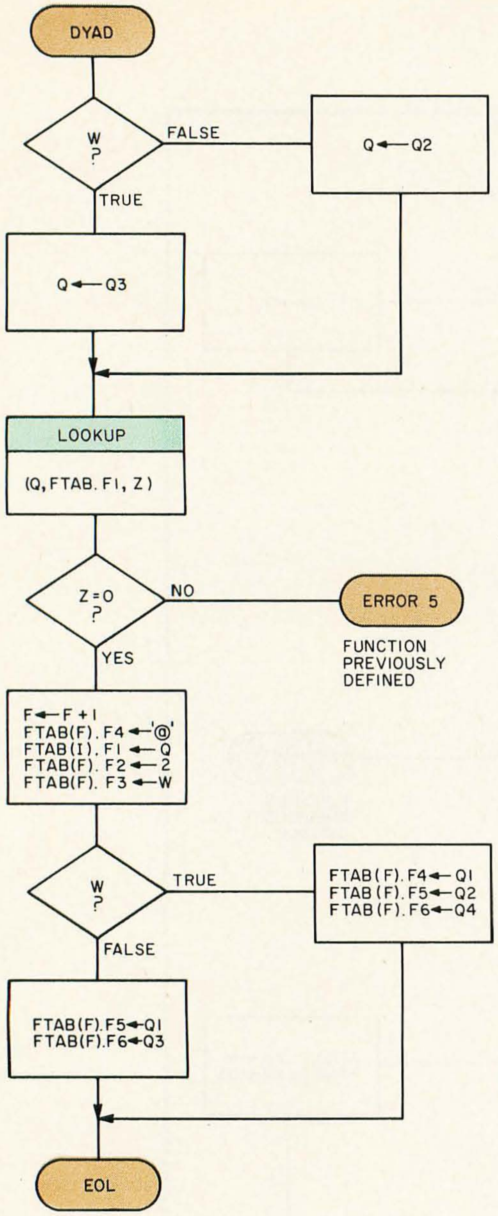


Figure 7c: Subroutine DYAD. This subroutine scans dyadic function headers. A dyadic function is one which takes two arguments.

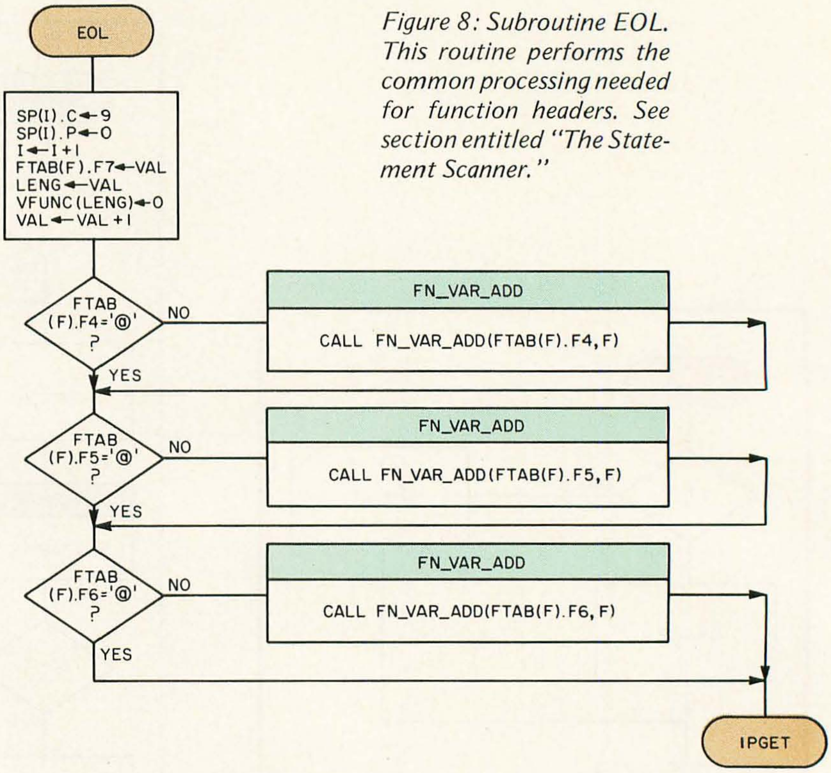


Figure 8: Subroutine EOL. This routine performs the common processing needed for function headers. See section entitled "The Statement Scanner."

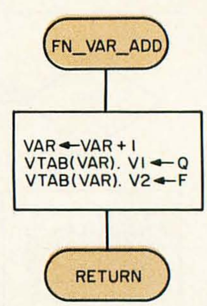


Figure 9: Subroutine FN_VAR_ADD. This routine aids subroutine EOL by adding user defined formal parameter names to the symbol table.

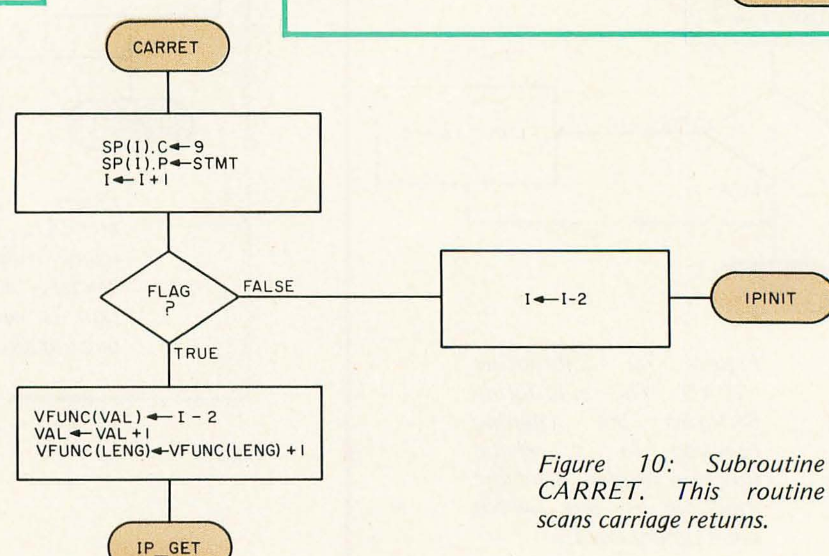


Figure 10: Subroutine CARRET. This routine scans carriage returns.

GLOSSARY

Array: A group of numbers or characters which can be subscripted and treated as a single object. APL arrays can be vectors or matrices.
Compiler: A program which translates a high level language source program into machine language object code. APL, like any language, may be compiled but is usually executed interpretively.
Composite Operator: An APL operator which operates on all the elements of an array.
Dyadic: Taking two operands. Used to describe certain APL functions or operators.
Execute Mode: The APL mode in which a line of APL code entered by the user is immediately executed by the computer (also called immediate mode). Compare *Function Defintion Mode.*
Function Definition Mode: The APL mode in which functions are created.
Function Header: A line of APL consisting of a del operator (∇) followed by a function name and information about the number of arguments the function takes and whether or not it returns a result.
Global Variable: A variable which is available to any calculation or any function in the workspace.
Interpreter: A program which takes a source listing from a high level language (like APL) and executes it sequentially (usually line by line).
Local Variable: A variable which is available only during the execution of the function in which it is defined. For example, in the function header

$$\nabla Y \leftarrow R\ Q;I;J$$

I and J would be local variables.
Monadic: Taking one argument. Used to describe certain APL functions or operators.
Niladic: Taking no arguments. Used to describe certain APL functions or operators.

REFERENCES

John Coake and J T Schwartz, *Programming Languages and Their Compilers,* Courand Institute of Mathematical Science, New York, 1970, pages 601-642.

Leonard Gillman and Allen J Rose, *APL 360: An Interactive Approach,* Wiley and Sons, New York, 1974.

IBM APL/1130 Primer, Student Text, IBM Publications, White Plains NY 1969. Highly recommended.

Kenneth Iverson, *A Programming Language,* Queens University, Kingston, Ontario, 1968.

Harry Katzan, *APL Programming and Computer Techniques,* Van Nostrand Reinhold, New York, 1970.

Harry Katzan, *APL User Guide,* Van Nostrand Reinhold, New York, 1971.

Sandra Pakin, *APL/360,* Science Research Associates, Chicago, 1972. A manual for the IBM 360 implementation of APL.

Sandra Pakin, *APL: A Short Course*, Prentice-Hall, Englewood Cliffs NJ, 1973.

Raymond P Palivka and Sandra Pakin, *APL: The Language and Its Usage*, Prentice-Hall, Englewood Cliffs NJ, 1975.

William Prager, *Introduction to APL,* Allyn and Bacon, Boston, 1971.

Xerox APL Reference Manual, Xerox Corp, El Segundo CA 1973. ■

Continued from page 5

philosophy course, the elements of this principle are:

> *Given multiple, equivalent logical formulations of an argument, that argument is best which is both complete and of the simplest structure.*
>
> *Complete means that the argument takes into account all the necessary details of the point being made.*
>
> *Simplest structure means that the argument in some sense is the most compact or easiest to understand.*

The whole point of this principle is that in order to reduce the complexity of long and involved logical constructions, one should choose paths of explanation which are direct and concise rather than long and convoluted. However, the principle says nothing about the need for complicated logical structures to describe complicated concepts. It merely says that for a given level of complexity the simplest argument, which accounts for all features of the point being made, is the best argument to adopt.

Simplicity and Personal Computing

The design of systems and the programming of computers are activities, like many human activities, which benefit from the application of this principle. If I take the liberty of reformulating the principle of simplicity from an engineering or programming point of view, it might read:

> *That system (program) is best which* accomplishes its goals *with minimum complexity.*

This application of an old principle to a modern context is a desirable design goal for the personal computer, or its attendant programming. But understanding and utilization of such constructs can only be simplified within the limits of the function which is the goal of the design. A complex structure is a complex structure, and ceases to be itself when pieces are removed by a simplification which compromises the goals and functions of the structure. Within any given definition of the functions or goals of a computing system or program, the principle of simplicity is most applicable; but if the overall goal is not simple, the available alternatives



for approaching this goal will necessarily reflect this lack of simplicity.

However, all is not lost. One of the greatest contributions of the past half century of systems engineering is what might be called a principle of local simplicity within a larger (ie: system wide or "global") structure. The goal is still complex, but the details are locally simple. This principle is inherited by today's personal computing designs, as well as much of modern technology from aerospace control systems to robotic manufacturing and process control.

The practical application of the concept of keeping designs and structures as simple as possible, consistent with accomplishment of complex goals, is seen in the idea of modularity as implemented in both hardware and software. It is nearly impossible to keep track of a myriad of details about a large and complicated system. But, if the design is approached from a modular and hierarchical point of view, the principle of simplicity is much more easily and usefully applied on a local basis and reflected in the entire system's conception. Local simplicity is used by the designer to minimize complication by choosing and evaluating the best alternatives at each level (or point of view) within the new system. It can be used when acquiring knowledge about a new area by the technique of slicing up a complicated subject into manageable units. We see the success of this method as design discipline in numerous ways, ranging from the modular low level hardware function boxes called integrated circuits, to the design of finished product personal systems, to the modular design of systems software, to the applications software an intelligent user designs modularly with an eye towards the problems of managing complexity.

Applying the Principle of Simplicity: A Caution

The owner of the personal computing system can use this principle of simplicity in numerous ways. However, it is not possible to accomplish positive goals with a personal computer by ignoring complexity and demanding some form of "absolute" simplicity which amounts to a lack of awareness of the subject matter.

The computer as a servant to mankind cannot be used to its fullest potential without active thought on the part of an interactive user. A "brain amplifier" is less than maximally useful without



brain activities (thoughts) to "amplify." With an understanding of the idea that most complex problems can be organized and presented by applying the principal of simplicity locally, a knowledge of the organization and design of systems and their use can grow and flourish in each active user's understanding of the technology.

I consider it very important to emphasize the point that the user of a computer is an active participant in the technology, that knowledge of the technology and its potential is essential to its full utilization by individuals. When we publish an article on some subject, it helps in this endeavor by providing a local and specific focus on that subject within the global knowledge of computer science. This is a sort of modularization of some aspect of the technology, with various pointers and references to the more general context of computer science practices as necessary. But each such element assumes an active and intelligent reader not willing to turn off input activities at the scanning of the end mark (■) for an article. There is no simple shortcut to knowledge and understanding; we can only provide aids and pointers to help mark the path.

Practical Philosophy for the Neophyte

The osmosis theory of education, coupled with a bit of inspiration from the principle of simplicity, is perhaps the best method of approaching the unknowns of personal computing systems on the part of the beginner. The osmosis theory may be described as a form of directed Brownian motion (in analogy to a well-known physical phenomenon of random motion). It is a directed activity in the sense that there is a particular goal used to judge the relevance of sources of information. It is a random activity in the sense that prior to acquiring an understanding of the field, there is no hard and fast criterion upon which to judge the order and timing of materials to be read and intellectually digested. In its simplest form, it is to simply plunge into the literature of a field and read every scrap of information which is related to the desired goal of understanding the subject. As new points are encountered with this approach, they are integrated into a personal internal structure of knowledge. The strategy is eminently successful when it covers all levels of detail about a subject through various readings, coupled with reflection and a bit of feedback which helps

eliminate the randomness in the direction of future readings.

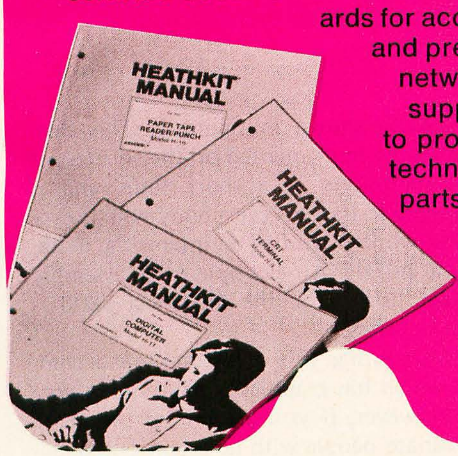
In one respect, I write about this osmosis method of acquiring knowledge (and its relationship to the principle of simplicity) with some authority; I know that the method is practical because I've personally applied it to gain my present understanding of computer science, such as it is.

This process of acquiring knowledge about a field is prone to an apt computer science analogy: imagine that the person acquiring the knowledge is analogous to a compiler program, that the source text of the compilation is the literature of the field, and that the object text out of the compiler is an internal understanding of the field. The analogy is illuminating with respect to the problems compilers often have with parsing and detecting subtle semantic nuances of a source language, or the problems of forward references to undefined elements of a program, which are analogous to gaps in the knowledge of a person embarking upon a detailed study of a field.

Learning About Mental Amplifiers

I think it can be safely asserted that the raison d'etre for computing is to amplify one's mental powers by automating various processes which can benefit from the effort. There is, of course, the intoxication with technology's details, which affects us all, and a certain degree of speculation on the experimenter's part as to whether an ultimately useful result will ensue when enough hardware and software development has been accumulated. But the pot of gold which attracts a lot of such speculations in the present era is the idea that computers can become amplifiers for mental functions. When both the mental and physical functions are modeled and automated, the result is a robot, hence one of the reasons for the dual fascination with robots and computers.

Mental powers and mental operations are far from simple. The range of activities to which I can turn my conscious attention is wide and varied; the same applies to every reader. In dealing with a complex subject, mental operations and their assistance through automation, the principle of simplicity is by no means to be forgotten, for it gives a criterion for judgment and discrimination among numerous alternatives at all levels.■



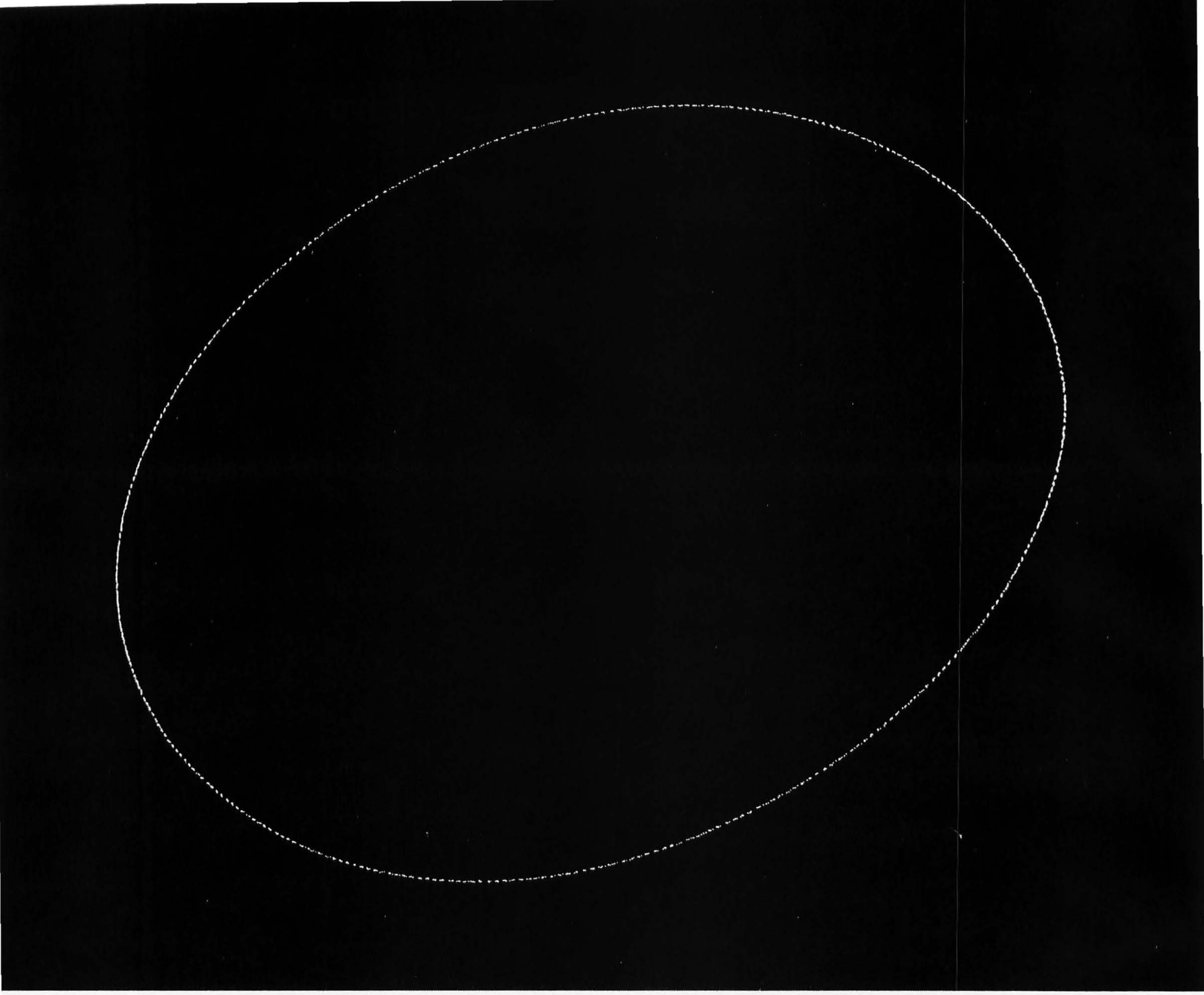


Photo 1: The basic ellipse created with the algorithm in figure 1 and with the origin centered. There is no evidence of "display wrap around" or arithmetic overflow here.

Serendipitous Circles

D John Anderson
Dept of Zoology NJ-15
Univ of Washington
Seattle WA 98195

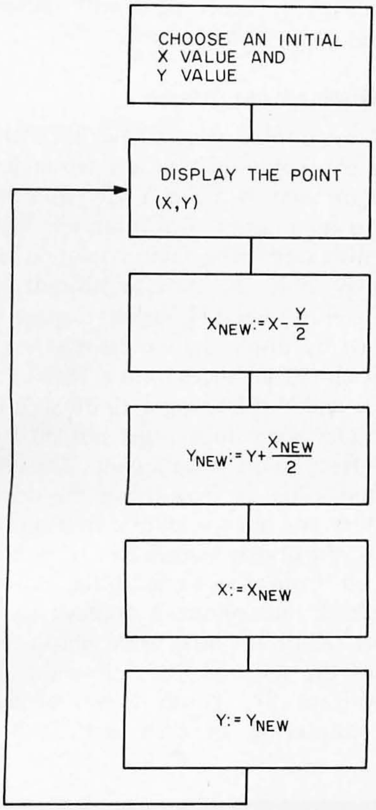
William F Galway
2246 Windsor St
Salt Lake City UT 84106

In this article we describe a simple algorithm which produces beautiful computer art. The algorithm was written to draw circles, but when we tried to draw large circles, we saw strange complex patterns. The screen was filled with wildly oscillating jelly bean shapes which would periodically explode in a beautiful sparkling flash and collapse back into new jelly beans. Experimenting with different diameters, we discovered many new dynamic patterns.

The Magical Algorithm

The algorithm that produced these serendipitous circles is remarkably simple. A flowchart is shown in figure 1. This algorithm can be implemented on the smallest of home computers in a short time, and is guaranteed to impress everybody with pretty pictures while providing an interesting mathematical puzzle. An example of an assembly language listing for the Motorola 6800 is shown in listing 1.

Implementing this algorithm on a computer which has graphics capabilities is very easy. However, if you are one of the many unfortunate people with no graphics capabil-



$X_{NEW} := X - \frac{Y}{2}$

$Y_{NEW} := Y + \frac{X_{NEW}}{2}$

Figure 1: The magical circle drawing algorithm. This straightforward procedure results in some surprisingly sophisticated displays when each coordinate pair is calculated and displayed on a graphics peripheral device. The algorithm solves a difference equation which has as its solution an ellipse. The nonelliptical patterns arise from overflows in the computer's registers during the calculations, and from lack of precision due to the finite size of computer words.

ities, we suggest the following possible solution: Borrow your friend's oscilloscope (the one you used to get your computer working). This oscilloscope should be able to draw a point on its screen, given an X deflecting voltage and a Y deflecting voltage. Now all you do is connect a digital to analog converter (DAC) to each of two parallel outputs from your computer, and you'll have the X and Y voltages necessary to drive the oscilloscope. This is how the 6800 program in the listing works.

The reason this program produces its beautiful patterns is quite well understood. The algorithm solves a difference equation which has as its solution an ellipse. The continuous analog of this equation is a differential equation which has as its solution a circle. The complicated nonelliptical patterns come from overflows and lack of precision due to the finite size of a computer

Listing 1: The serendipitous circle algorithm as written for the 6800 microprocessor. This program uses 16 bit arithmetic. The photos shown in this article illustrate circles generated with 24 bit arithmetic, although the results for both methods are quite similar. Different precision arithmetics can be used to generate slightly different results.

```
                              NAM     CIRCLE
                              OPT     M
                         *******************************************
                         *
                         *CIRCLE DRAWING PROGRAM
                         *WRITTEN BY WILLIAM F. GALWAY
                         * 30-DEC-76
                         *THE ALGORITHM USED IS AS FOLLOWS:
                         *
                         *LOOP:  DISPLAY A POINT AT CURRENT (X,Y)
                         *         X <- X-Y/2
                         *         Y <- Y+X/2
                         *         GOTO LOOP
                         *
                         **********************************************
                         *
                         *SIXTEEN BIT ARITHMETIC IS USED ALTHO ONLY
                         *EIGHT BITS ARE DISPLAYED.
                         ****************************
                         *
                         *SOME GOOD STARTING VALUES FOR (X,Y) ARE:
                         * (7F00,0000)
                         * (7F00,8100)
                         * (7D00,7D00)
                         * (7CF3,7CF3)
                         *
                         ****************************
                         *DEFINE  SOME SYMBOLS
            3B0B         WK0     EQU     $3B0B      WORK REGISTERS, USEFUL FOR
            3B0D         WK1     EQU     WK0+2      GETTING NUMBERS FROM
            3B0D         XHI     EQU     WK1        WORK REG'S USED FOR X,Y
            3B0F         WK2     EQU     WK1+2
            3B11         WK3     EQU     WK2+2
                         ***************************************
                         *DEFINE ADDRESSES FOR
                         *DIGITAL TO ANALOG CONVERTERS
                         *THE FOLLOWING THREE REGISTERS ARE
                         *UNIQUE TO THIS SYSTEM.
                         ***************************************
            3804         DACCSR EQU      $3804      USED FOR SETTING UP DAC'S
            3805         DACX    EQU     DACCSR+1 DAC FOR X VOLTAGE
            3806         DACY    EQU     DACX+1     DAC FOR Y VOLTAGE
      2000                       ORG     $2000

      2000 CE 3B0D START LDX     #XHI       POINT TO HI ORDER X
                         *THE FOLLOWING TWO INSTRUCTIONS ARE
                         *UNIQUE TO THIS SYSTEM, AND ARE NECESSARY
                         *TO MAKE THE DAC'S WORK IN THE DESIRED MANNER.
      2003 86 30                 LDA A   #$30       SET UP DAC CSR
      2005 B7 3804               STA A   DACCSR     TO SHOW DATA NOT ADDRESS
      2008 A6 00       LOOP      LDA A   0,X        GET HI ORDER X
      200A E6 02                 LDA B   2,X        GET HI ORDER Y
      200C B7 3805               STA A   DACX       DISPLAY A POINT AT (X,Y)
      200F F7 3806               STA B   DACY
      2012 E6 03                 LDA B   3,X        B GETS LO Y
      2014 A6 02                 LDA A   2,X        A GETS HI Y
      2016 47                    ASR A              GET Y/2
      2017 56                    ROR B
      2018 40                    NEG A              DO A 16 BIT NEGATE
      2019 50                    NEG B              NEGATE LO ORDER
      201A 82 00                 SBC A   #0         PROPAGATE CARRY
      201C EB 01                 ADD B   1,X        GET X+(-Y/2) LO ORDER
      201E A9 00                 ADC A   0,X        THEN HI ORDER
      2020 E7 01                 STA B   1,X        STORE LO X
      2022 A7 00                 STA A   0,X        STORE HI X
      2024 47                    ASR A              GET X/2 HI ORDER
      2025 56                    ROR B              LO ORDER
      2026 EB 03                 ADD B   3,X        ADD LO Y
      2028 A9 02                 ADC A   2,X        ADD HI Y WITH CARRY
      202A E7 03                 STA B   3,X        SAVE LO Y
      202C A7 02                 STA A   2,X        SAVE HI Y
      202E 20 D8                 BRA     LOOP       LOOP TO DISPLAY SOME MORE!!
                                 END
```

The screen was filled with wildly oscillating jelly bean shapes which would periodically explode in a beautiful sparkling flash and collapse back into new jelly beans.

word. For example, overflow occurs when your computer adds two positive numbers and gets a negative number. Lack of precision occurs when your computer divides 5 by 2 and gets 2 as the result.

Variations on the Algorithm

There are several different ways that the program in figure 1 can be varied. Some of these are described below.

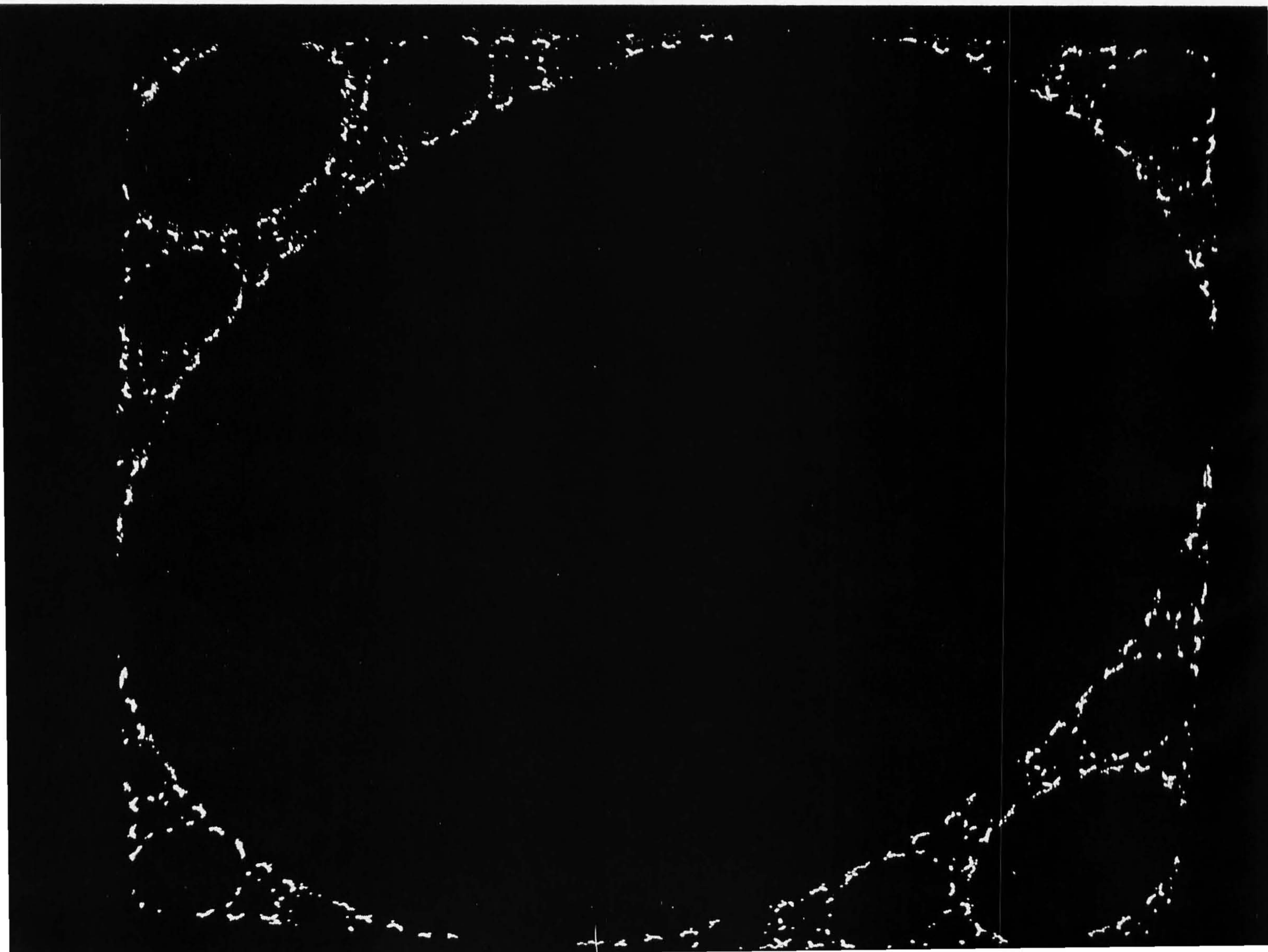
- **Initial X and Y**

The choice of a starting point determines the pattern the program draws. Points close to the origin draw ellipses; larger values can produce more interesting patterns. Photo 2 was produced by using slightly larger starting values than for photo 1. The best way to understand this effect is to experiment with different values of X and Y yourself.

- **Display Wrap Around**

Most display devices have less than 16 bits of resolution, typically ten or fewer bits are used. If X and Y are represented as 16 bit integers, which ten bits should be displayed? The logical solution is to display only the most significant bits. Different interesting results can be obtained by displaying successively less significant 10 bit slices from a 16 bit value of X and Y. Choosing a 10 bit slice that excludes some high order bits produces an effect called wrap around. The effect is the same as magnifying the original picture and then folding it over on itself to fit the display screen. Photo 3 was produced from the same initial point as photo 2; but photo 2 displays the ten most significant bits, while photo 3 displays the ten bits just below the most significant bit. Photo 4 was produced by displaying an even less significant

Photo 2: Effects created by overflows and lack of precision of finite computer words. The initial point (X, Y) for this photo was chosen to be slightly further from the origin than the initial (X, Y) used for photo 1.



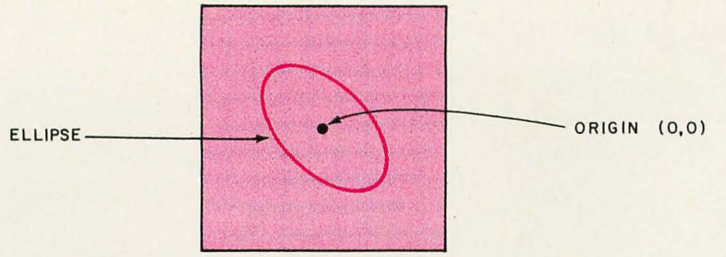


Figure 2a: One of two preferred locations for the origin of your coordinate system display. Here the origin is placed in the middle of the display to produce an ellipse centered about the origin. See photo 1.

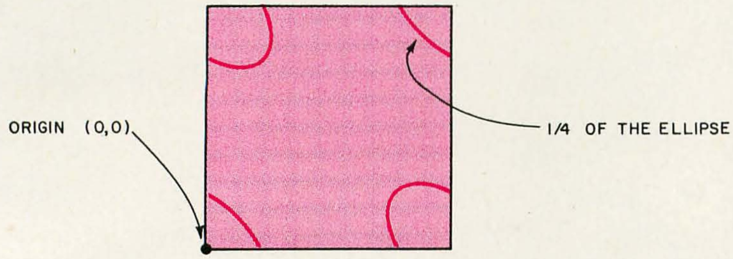
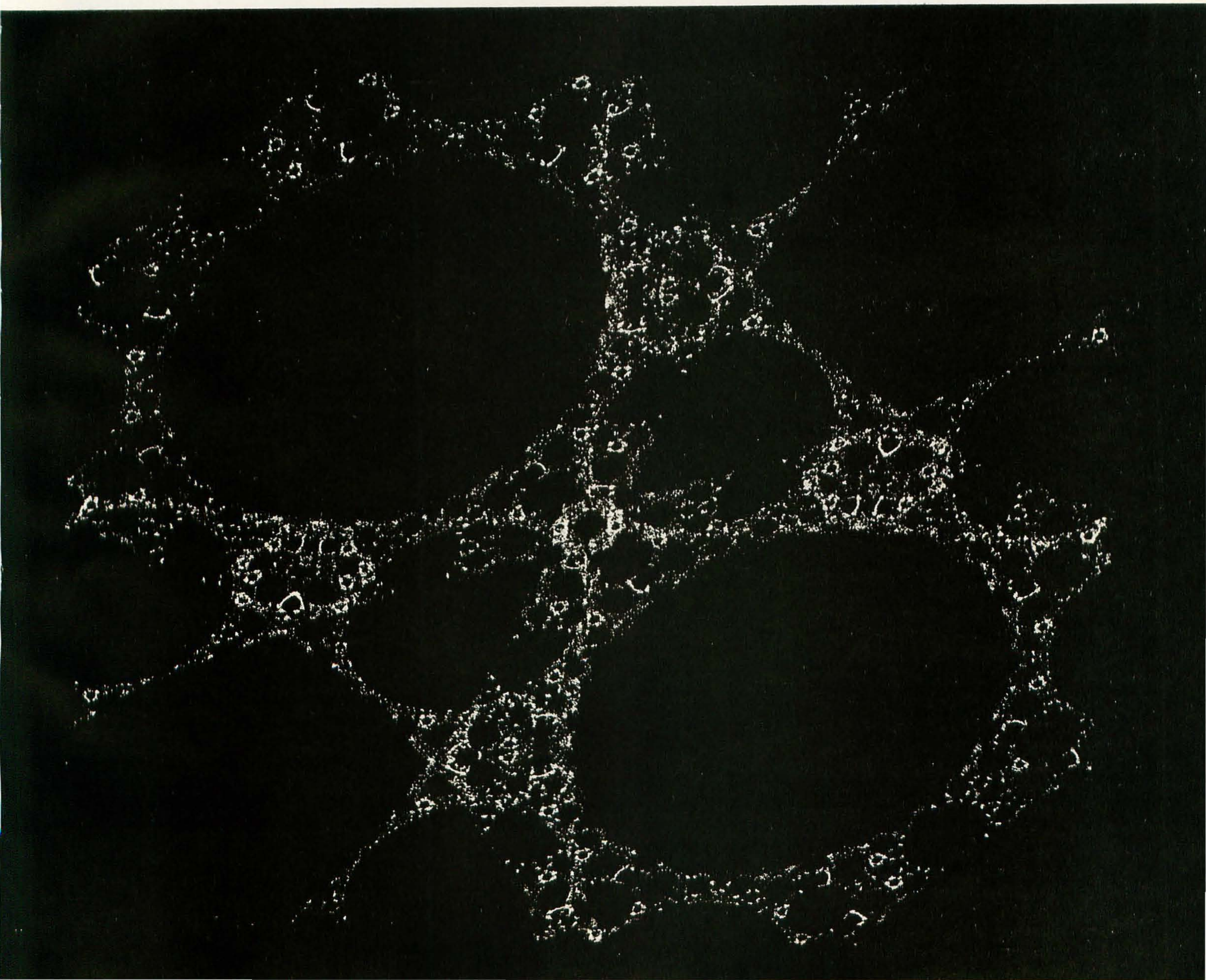


Figure 2b: In this case the origin has been placed in the corner, splitting the ellipse up into four quarters. This is called "display wrap around."

Photo 3: This photo shows the effect of displaying the ten bits just below the most significant bit for both X and Y. Note the radical change in appearance.



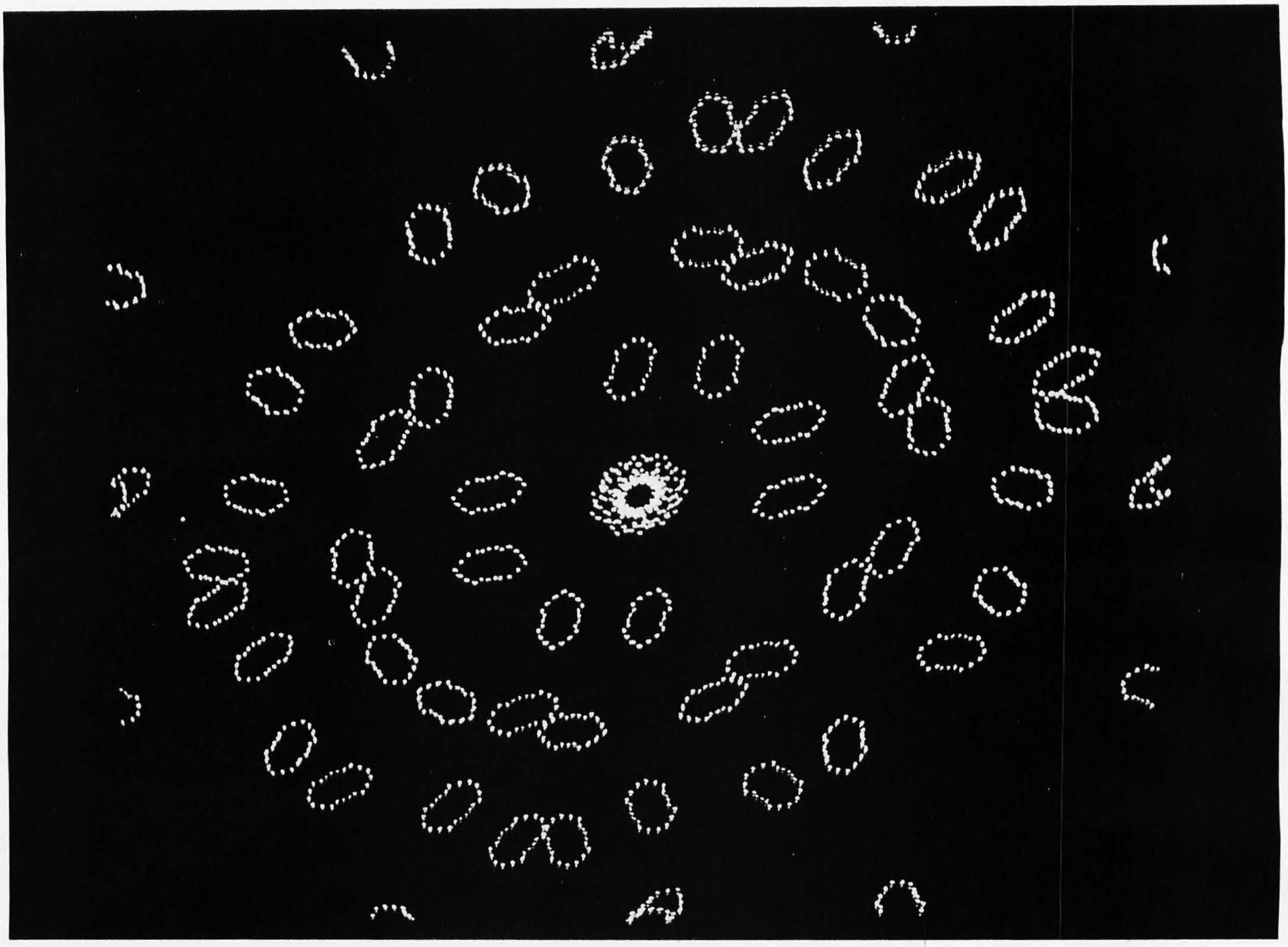


Photo 4: Once again the display changes significantly, this time by choosing an even less significant group of ten bits for display. The technique of using only certain groups of bits is analogous to magnifying the original picture and then folding it over on itself to fit the display screen.

group of ten bits and using a different starting point.

- **Position of Origin**

Figure 2 shows two choices of placement for the origin. Choosing the origin in the center of the display, as in figure 2a, results in the ellipse drawn in the center of the display. However, if the origin is placed in the corner as shown in figure 2b, then one quarter of the ellipse is drawn around each corner. Choosing the origin in the center corresponds to using a signed digital to analog conversion and choosing the origin in the corner corresponds to an unsigned conversion. Patterns that are not very interesting with one origin may be very interesting with the other origin.

- **Speed of Display**

The speed of displaying points, ie: the number of points per second, is an important factor in the appearance of a pattern. You can put a loop in your program to introduce a delay and experiment with this effect.

Generalizing the Equation

Still more patterns can be produced by using the more general equation:

$$X_{NEW}:=X - aY$$
$$Y_{NEW}:=Y + bX_{NEW}$$

About the Authors

William F Galway is employed by the University of Utah working with computer-aided instruction using computer graphics. It has been known by many people that the algorithm described in this article will draw ellipses and circles. While working with a computer music research group at the University, Mr. Galway noticed the curious behavior of the algorithm when large diameters were used.

D John Anderson is a graduate student at the University of Washington in the Department of Zoology working on difference equation models of population growth. In the past, he was employed at the University of Utah where he worked with minicomputers in physics and chemistry.

The photographs in this article used a=b=½. We have tried values other than ½ with interesting results, but have never used different values for a and b.

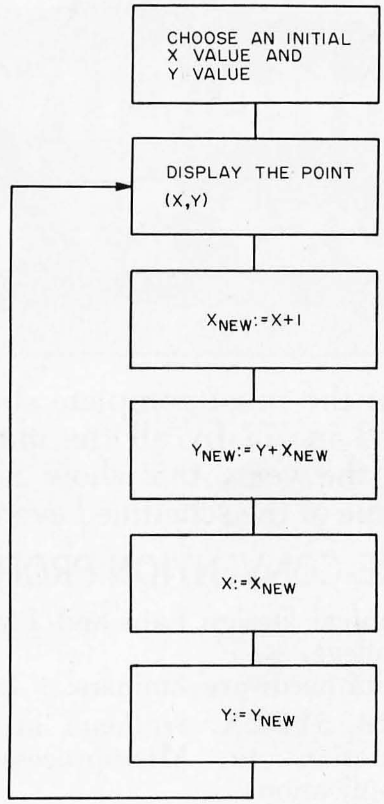
Precision of Arithmetic

Most of the programs we have written to draw our patterns have used 16 bit arithmetic. The program used for the pictures in this article used 24 bits (double precision on a PDP-8 computer). It is possible that using fewer bits will not produce such interesting results.

Other Algorithms

Another algorithm you might try is shown in figure 3. Without much trouble it can be shown that this is a difference equation analog of a differential equation that draws parabolas. This particular algorithm can produce interesting and complicated musical sounds if the output of the Y DAC is amplified and played through a speaker.

It is easy enough to try new forms of these or other difference equations. It has been our experience that these algorithms often produce new visual and audio effects that will astound you.■



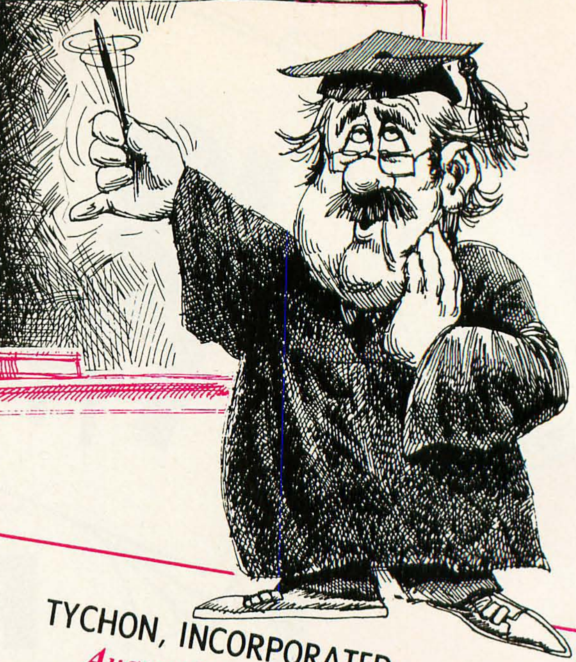
This algorithm can produce interesting and complicated musical sounds if the output of the Y digital to analog converter is amplified and played through a speaker.

Figure 3: A serendipitous algorithm based on a parabola. (This can be more easily seen by manually executing the algorithm and plotting a few points for X and Y.) The parabola is the solution to a difference equation which forms the basis of the algorithm.


PC
77

Personal Computing College™

In-depth information through seminars, lectures and workshop sessions covering the personal computer field.



OSBORNE AND ASSOCIATES, INC.

August 26 & 27

"MICROPROCESSORS — WHERE THEY CAME FROM AND WHERE THEY ARE GOING. AN ANALYSIS OF ALL PRODUCTS ON THE MARKET TODAY." Dr. Adam Osborne will present a 6 hour seminar covering topics such as; Bringing order out of chaos; All microprocessors are not equal — each serves one market better than the other; Identifying those markets best suited to each microprocessor; Real sales volume anticipated for 1977; Comparisons including; 8085 vs. Z80, 8048 vs. F8, Etc. For information and reservations contact: Osborne and Associates Inc., Dept. PC77, P O Box 2036, Berkeley, CA 94702 (415) 548-2805.

TYCHON, INCORPORATED

August 24, 25 & 26

3 DAY MICROCOMPUTER INTERFACING WORKSHOP

A hands on experience for the participants where they will spend almost 50% of their time working on well documented Interfacing and Software experiments. Students deal with the microcomputer at the bus level, interfacing the computer using solderless breadboarding techniques and assembly language programs. Presented by Jonathan A. Titus and Dr. Christopher A. Titus, Authors of the famous "BUGBOOKS." For information and reservations contact: Tychon, Inc., c/o Shortness-Rawson, Dept. PC77, P O Box 2203, South Hackensack, NJ 07606.

SYBEX INCORPORATED

Thursday, August 25 & Sunday, August 28

INTRODUCTION TO MICROPROCESSORS

This intensive seminar is intended for all non-specialists who wish to acquire a broad understanding of the basic concepts and advantages of microprocessors. It explains how microprocessors work and it stresses methods, costs, advantages and disadvantages for the most important application areas of each type of microprocessor. What is needed to implement a system; how to use it; the impact on microprocessor-based systems; their evolution. Topics covered include: BASIC DEFINITIONS, SYSTEM COMPONENTS, MICROPROCESSOR APPLICATIONS, WHAT TO LOOK FOR, and IMPACT AND EVALUATION.

PROGRAMMING MICROPROCESSORS

This seminar describes in detail the internal operation of a microprocessor system including how instructions are fetched and executed, how programs are written and executed in typical cases (arithmetic and input-output). The goal of this course is to provide an overall understanding of the basic concepts of microprocessor programming. Requires an understanding of the main concepts in the INTRODUCTION TO MICROPROCESSORS course. It is recommended that these two seminars be taken together.

Friday, August 26

MICROPROCESSOR APPLICATIONS

This seminar presents in detail the main application techniques of microprocessors. Topics covered include INTRODUCTION TO MICROPROCESSOR SYSTEMS, APPLICATION TECHNIQUES, CASE STUDIES (industrial applications, medical and business applications, microprocessors in the home, and others), and EVOLUTION. For information and reservations contact: Sybex Inc., 2161 Shattuck Ave., Berkeley, CA 94704 (415) 848-8233.

TECHNICAL DESIGN LABS AND TRENTON STATE COLLEGE

August 22 through 26

PROGRAMMING IN BASIC FOR THE uC OWNER.
An introduction to the Basic language. From beginning to writing application programs. Emphasis on TDL's 8K and 12K Basic for the Z80. Includes intro to TDL's FORTRAN.

ASSEMBLY LANGUAGE PROGRAMMING FOR THE Z80/8080
An introduction to assembly language programming. First covers 8080 instructions and then branches into the extra instructions available for the Z-80. Includes intro to TDL's Z-80 Monitor and Macro-Assembler and also covers applications in interfacing and control.

ADVANCED ASSEMBLY LANGUAGE PROGRAMMING FOR THE Z-80.
For someone who knows the 8080. Covers added Z-80 instructions and how to get the most out of them. Emphasis on use of TDL's Macro-Assembler. Application to digital logic replacement and process control.

FORTRAN IV
An introduction to the FORTRAN language. From beginning to writing of application programs. Emphasis on TDL's ANSI standard FORTRAN IV for the Z-80.

WORD PROCESSING WITH A TEXT EDITOR AND FORMATTER
Introduction to the use of the Text Editor and Text Output Processor for achieving basic word processing capabilities. Emphasis on TDL's Text Editors and Text Output Processors. Familiarization with system requirements.

INTRODUCTION TO HOBBY COMPUTING
A survey course dealing in an exploration of the Hobby computing field and defining the basics needed to be understood in order to get started.

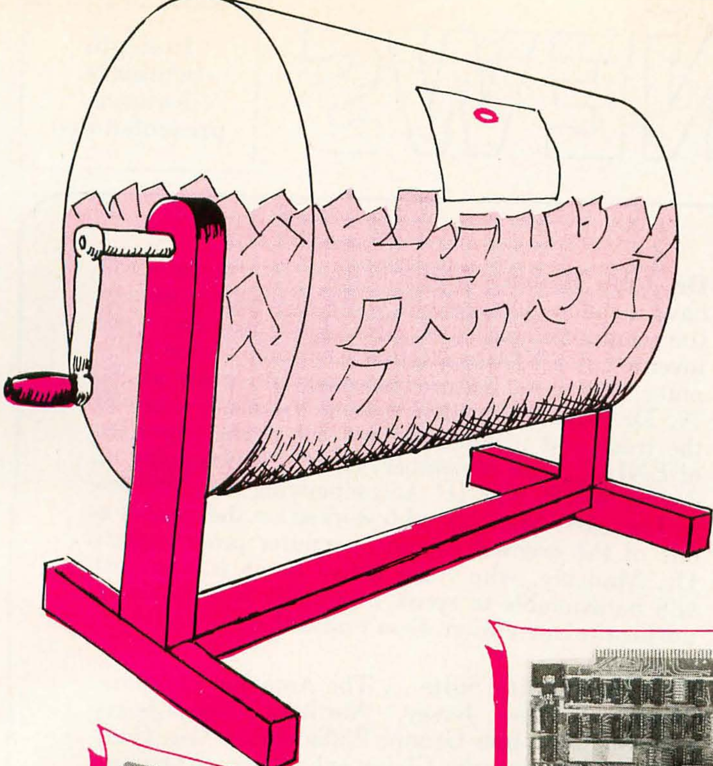
DIGITAL LOGIC CIRCUITS
Instruction in digital logic circuits covering the 7400 TTL and the CMOS series. Codes, registers, counters, memory, combinatorial logic, etc.

HOW TO SET UP A COMPUTER STORE
Guest lectures from owner/operators of computer stores and microcomputer manufacturers. How to become a dealer. How to get a franchise. How to operate a business. How to set up a service facility . . . and more.

KIT BUILDING LAB
Instruction and guidance on kit building. Bring your kits! SPECIAL: Anyone taking delivery on a TDL product during the course (must be ordered in advance) will receive this workshop FREE.

For information and reservations contact: Z-80 Seminars, Office of Continuing Education, Trenton State College, Trenton, NJ 08625.

OSBORNE ASSOCIATES, SYBEX AND TYCHON SEMINARS ARE ALL HELD IN THE SHELBURNE HOTEL, ATLANTIC CITY, NEW JERSEY. TECHNICAL DESIGN LABS AND TRENTON STATE COLLEGE SEMINARS ARE HELD AT NEARBY TRENTON STATE COLLEGE, TRENTON, NEW JERSEY.



DOOR PRIZES

★★★★★★★★★★★★★

COME SEE

★★★★★★★★

MORE

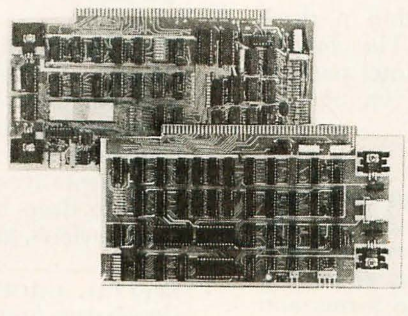
MORE

MORE



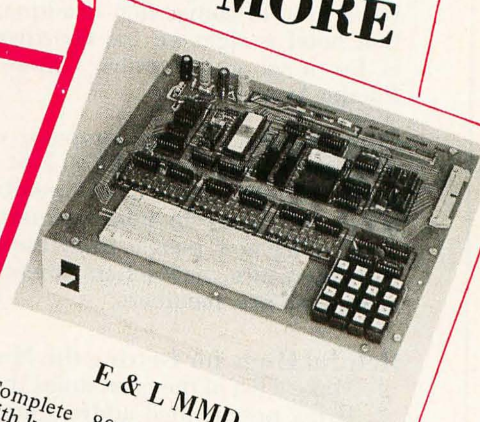
Midwestern Scientific Instruments

The newest and most advanced 6800 Computer System available. At PC '77, MSI will demonstrate the new MSI 6800 and the MSI FD-8 Floppy Disk Memory System running Multi-User BASIC with full program and Data File handling capability.



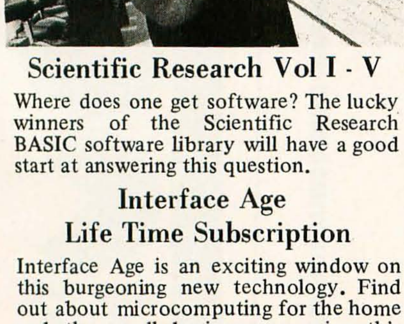
Solid State Music

Plug in any one of a number of Solid State Music modules and enhance the power of an Altair (S-100) bus system. In particular enjoy the new SB-1 music synthesizer and the VB1A video board to be given out as door prizes at PC77.



E & L MMD - 1

Complete 8080 based microcomputer with keyboard, 500 word RAM memory, 24 LED read out, power supply and case. Includes Bug Books V & VI, programmed self-training course in digital electronics and microcomputer technology.

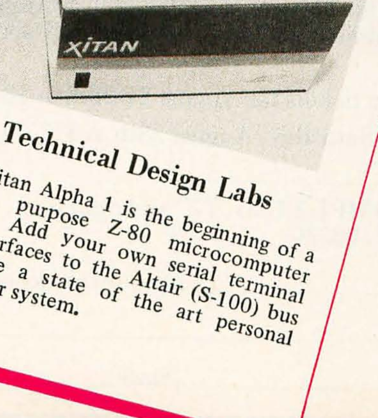


Scientific Research Vol I - V

Where does one get software? The lucky winners of the Scientific Research BASIC software library will have a good start at answering this question.

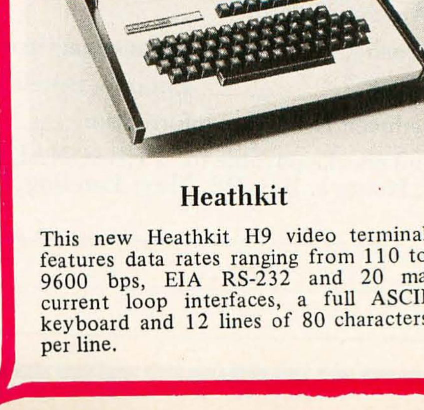
Interface Age Life Time Subscription

Interface Age is an exciting window on this burgeoning new technology. Find out about microcomputing for the home and the small businessman using this monthly compendium.



Technical Design Labs

The Xitan Alpha 1 is the beginning of a general purpose Z-80 microcomputer system. Add your own serial terminal and interfaces to the Altair (S-100) bus to create a state of the art personal computer system.



Heathkit

This new Heathkit H9 video terminal features data rates ranging from 110 to 9600 bps, EIA RS-232 and 20 ma current loop interfaces, a full ASCII keyboard and 12 lines of 80 characters per line.

Register before August 10 SAVE 20%

PC '77 NEWS

In-depth Seminars, lectures, presentations!

Silicone Sea ... There's a new excitement in Atlantic City. With the advent of casino gambling $800,000,000 is being invested by such companies as Caesar's International, Playboy, Penthouse, Bally Corporation, Resorts International, Loew's Corporation and many others in new hotel, restaurant and entertainment facilities. And of course all the old attractions of this famous resort remain, the miles of sandy beach with gentle surf, the clean, well run amusement parks, the boardwalk with its exotic shops and much more. All of this adds up to the perfect family vacation spot, easily reached in less than two hours driving time from Washington DC, Philadelphia or New York City.
There is parking for 20,000 cars within a three block area of convention headquarters. The convention hotel is right on the boardwalk. Hotel parking is for hotel guests only. Others park on near-by Pacific Avenue.

Club Booth ... Every computer club is welcome to participate with us at PC 77 at our gigantic club booth. Clubs may have members manning the booth to engage in non-commercial activities such as meeting new people from their areas, giving out club literature, newsletters, membership forms and signing up new members.

Dr. John Mauchly and ENIAC ... Enthusiasts will have a chance to meet one of the earliest pioneers of the computer industry. Dr. John Mauchly, co-inventor of ENIAC, the first electronic digital computer, will be a featured speaker and guest at PC 77. Dr. Mauchly will be telling the inside story of the trials and tribulations, as well as the triumphs, of ENIAC. And remember, all of this was going on during World War II! Accompanying Dr. Mauchly will be Mrs. Mauchly who worked on the project as one of the original woman computer programmers. Dr. Mauchly, who is an active home computerist, will be available to speak informally with hobbyists during the convention. **Don't miss this!**

Club Hospitality Suite ... The Amateur Computer Group of New Jersey, Northern New Jersey Amateur Computer Group, Philadelphia Area Computer Society and the Chesapeake Micro Computer Club will host a hospitality suite during the show. Representatives of all computer clubs are encouraged to drop by the suite during the show to exchange views, greet old friends and meet new ones.

SPECIAL GROUP TRAVEL RATES ... for Clubs and Organizations from West Coast and Mid America. Contact Seven Seas Travel, 17220 South Norwalk Bldv., Cerritos, Calif. 90701, Dawn Corrigan, 213/924-8383.

Helpful Hints for Getting the Most Out of the Show

1. Make a list of the companies that you particularly want to see.
2. Bring pregummed address labels with your address printed on them for exhibitor questionnaires, booth prize tickets and requests for information. This will save having to write your name hundreds of times and give you more time to get around the show.
3. Plan on returning to your room or car several times during the day to drop off literature and to change into a fresh pair of comfortable shoes.
4. Bring a good tape recorder, lots of spare batteries and tapes and record information, make notes, and record the seminars. (Recording of seminars is for personal use only, reproduction is prohibited.)
5. If you are planning to attend only one day, make that day Saturday. If you find you really need more time you'll be able to come back on Sunday.
6. Make hotel reservations early. Have them confirmed to save both time and disappointment.
7. Bring spare pencils, pens, and note paper.
8. Bring adequate identification in order to make purchases.
9. Be open minded when talking to exhibitors. Listen to what they have to say, you may learn something valuable.
10. Bring your camera, there will be much to photograph.
11. Remember that **advance registration** will save you from having to wait in line. Mail the coupon below today!

PC '77 Weekend admission at the door will be $10.00. Register before August 10th SAVE 20% AVOID WAITING IN LINE! Admission includes exhibits and seminars for both days, August 27-28th.

Please rush ___________ advance registration tickets for August 27-28th at $8.00 each.

Please rush ___________ Banquet tickets for Saturday, August 28th at $17.50 each.

Send reduced Hotel rate information ______________________.

Make all checks payable to "PERSONAL COMPUTING '77" and mail to:
PC'77, Route 1, Box 242, Mays Landing, N.J. 08330

Name ______________________ Amount enclosed __________

Address ______________________

City ______________________ State __________ Zip __________

PC 77

Editorializing with Your Computer

Gary McGath
7 Silver Dr, Apt 3
Nashua NH 03060

One of the basic requirements of a software system is a way to get text input. Programs need data to work on; and you need some way to turn this data from markings on a piece of paper to bits inside the machine. The most obvious examples, if only because you have to use them before you can have any other programs of your own creation, are assemblers and compilers. The source program to be translated is just the data for the translating program. Other kinds of programs, tabulations, statistical analyses, and so forth, also cry out for a convenient way to get a large amount of text into the computer. This is where the text editor comes in.

A text editor is a program which allows the user to enter and modify encoded character data. With it, you can enter a program or a list of figures or anything else, go back and make corrections, and finally output the text onto a storage device or leave it in memory for your next program to work over. A text editor is a fairly simple program to write, and the 8 bit machine which you are probably using actually has one advantage for editing over many big computers: Its unit of information is just the right size for holding one ASCII character.

There are two basic requirements for an editor: a place to put the text and a way to handle it. The place to put it is the text buffer, which is simply a block of main memory of a reasonable size. The way you handle it is with a command language, which will run interactively through your console.

The buffer can always have the same starting location, but where it ends depends on how much text it holds at any moment. So a pointer indicating the current end of the buffer has to be kept. Initially (or after you "scratch" your file), it points to the beginning of the buffer, since there is no text yet. The convention assumed here is that the end of buffer pointer indicates the first unused location. There is also an absolute end to the buffer, which does not change; this is the end of the memory allocated for it. The editor should guard against any attempt to store characters beyond this point, for that might wipe out a program or crucial data.

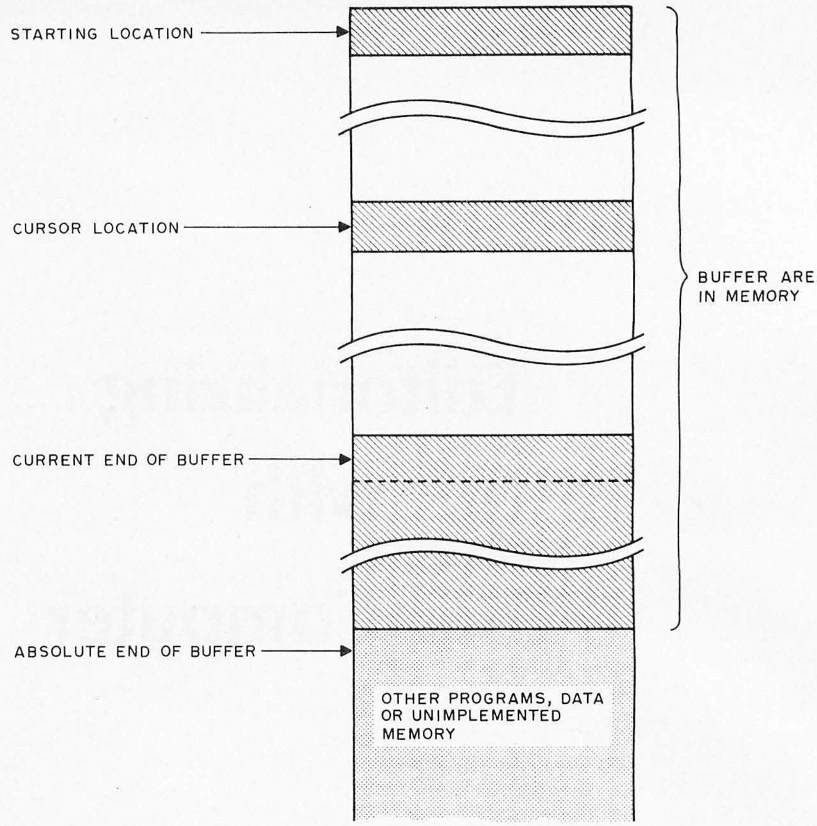
Another pointer which the editor needs is the cursor. This pointer indicates the exact location in the buffer which will be affected by editing commands. Inserted characters will appear at this point; deletions will wipe out the characters which are here. When the editor starts up, the cursor addresses the beginning of the empty buffer.

Text Modifying Commands

The commands which an editor needs can be classified into text modification, cursor

The Text Editing Buffer Concept

A text editing buffer is an area in memory address space which has programmable memory implemented. It is the computer equivalent of a blackboard on which to create programs, letters to friends, or even novels or other writings. A typical way of describing such a buffer is through several pointers (address values) which identify the limits of available programmable storage. The starting location is a fixed address which is built into the text editor program when such a program is typically implemented on a small computer. Similarly, the absolute end of buffer space is a fixed address which points to the next location after the end of the buffer. The cursor location is a variable in the text editing program which is allowed to range between the starting location value and the current end of buffer. The current end of buffer is another variable, which identifies the location of the next available unused byte in the buffer space. This variable should be constrained to be greater than the cursor location, but always less than the absolute end of buffer. When the current end of buffer is the last byte in the buffer, the buffer is full.



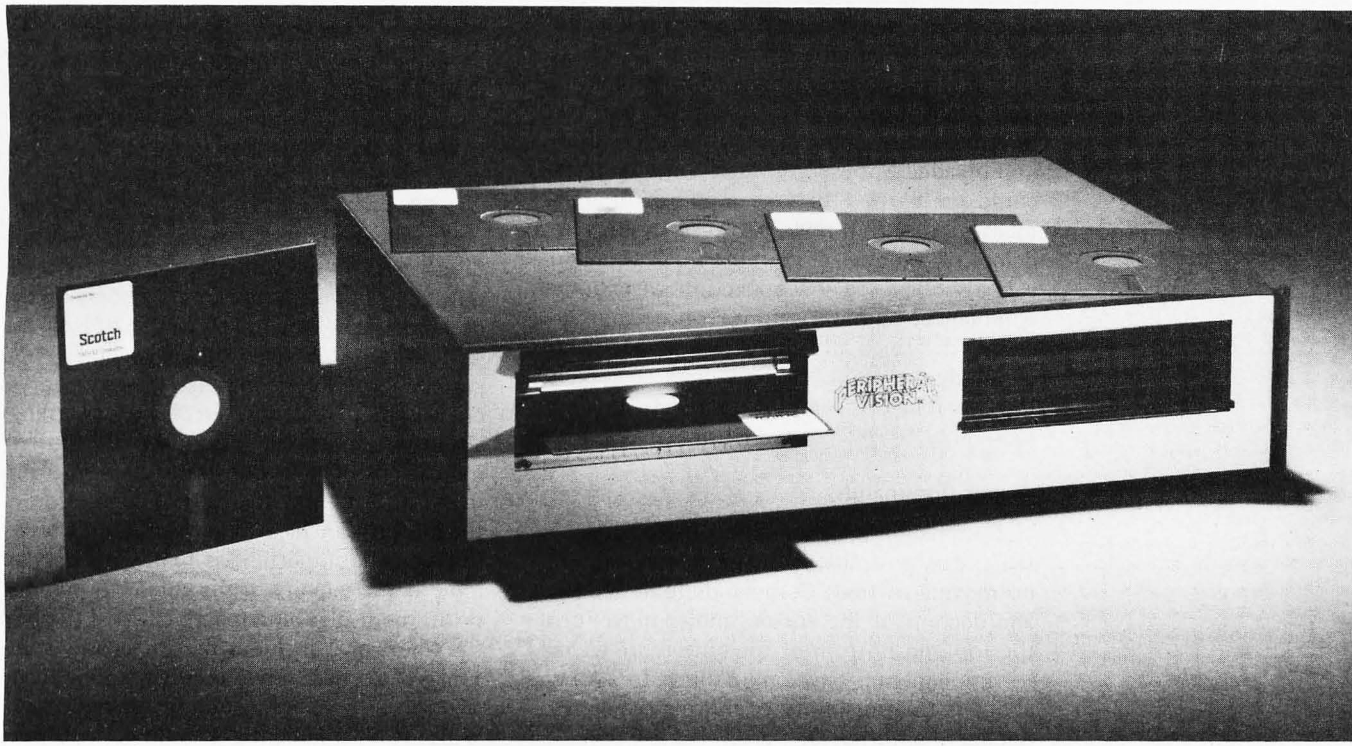
positioning and IO commands. Since the main concern of the editor is to do something with text, I'll start with the commands that modify text.

At the very least, a command to input text from the keyboard is needed. There doesn't actually have to be an explicit input command; if all commands are issued by special control characters (excluding carriage return, tab, etc), then the editor can take every noncommand character which it reads in its normal course and insert it directly into the text buffer. Alternatively, a specific input command can cause all text which is typed to be stored in a temporary buffer, up to a preselected termination character, such as the aptly named "escape" code. The temporary buffer is then copied into the text buffer after getting the termination character. The latter method makes typing in control characters, except the termination character, easier. It is even more efficient in some cases where "efficient" means quicker response to the user. On the other hand, direct insertion can be handled with fewer instructions, avoids the need for a buffer, and eliminates the chance of forgetting whether the editor is waiting for commands or input.

Inserting characters is easy when the cursor is at the end of the buffer. The characters are just deposited there, and the cursor is moved along. But when the cursor is somewhere else, then the character which it points at and all the characters after it have to be moved down to make room for the inserted character(s). This is where buffered input is more efficient. Without it, the characters have to be pushed down every time a new character comes in; with it, they can be moved over by the number of characters in the buffer all at once. If, for example, you want to insert 40 characters at a point in text that has 100 characters after it, you would have to do 4000 memory to memory transfers, not counting the insertion itself, without buffering, as opposed to 100 with buffering. But this only matters if you can type characters in faster than your processor can handle them. *[In the text edition I wrote for my 6800 based system, I used the "quick and dirty" brute force technique just described. With my homebrew machine on long text chunks (12 K or so) it takes typically 1/5 second for the program to respond, intolerably slow . . .* CH*]*

The other basic text modifying operation is character deletion. Probably the best character to delete is the one just before the cursor, so that you can rub out what you just typed in if you made a mistake. But be sure to check for deleting past the beginning of the buffer! If the deletion occurs at the

FLIP OVER OUR FLOPPY



Only $750 from Peripheral Vision.

Peripheral Vision is a brand-new company that's dedicated to selling reasonably priced peripherals for various manufacturers' CPU's.

We think you'll flip over our first product.

It's a full-size floppy disk for the Altair-Imsai plug-in compatible S-100 BUS. And it's available for as low as $750.

Here are the features:

- 1 interface card supports 4 drives
- Stores over 300,000 bytes per floppy
- Bootstrap EPROM included—no more toggling or paper tape
- Completely S-100 plug-in compatible
- Interface cabling included
- Drive is from Innovex (the originator of the floppy concept)—assembled and tested
- Interface card design is licensed from Dr. Kenneth Welles and the Digital Group
- Disk operating system with file management system included on floppy
- Cabinet and power supply optional

Prices:	Kit	Assm.
Interface card kit and assembled and tested drive	$750	$850
Power supply—+24V at 2A	45	65
Cabinet—Optima, blue	—	85

Now, a little more about our company.

Peripheral Vision may be brand-new, but we have some old-fashioned ideas about how to run our business.

We know there are serious incompatibilities among the different manufacturers' peripherals and CPU's. We want to get them together. And, we want to bring significant new products to market—products consisting of everything from adaptation instructions/kits for hardware and software to major new products.

It's a tall order, but we feel we're up to the task. Peripheral Vision has already obtained a license from The Digital Group to adapt versions of some of their products to the S-100 BUS. And we're working on getting more from other companies.

Most important to our customers, Peripheral Vision is committed to helping you get along with your computer. We'll do all we can to make it easy.

Write us now for all the information on our company, our philosophy and our exciting line of products. And be prepared to flip over all of it.

PERIPHERAL VISION INC.

P.O. Box 6267 / Denver, Colorado 80206 / (303) 733-1678

Send me the works, and I just might flip over it!

Name____________________

Address____________________

City/State/Zip____________________

end of the buffer, then the end of buffer pointer just has to be decremented; for neatness, the deleted character might also be zeroed out. If the cursor is in the middle of the buffer, then the character it points at and all subsequent characters are moved back one position, thus overwriting the unwanted character.

For added power, the deletion command could be allowed to take a numerical argument, indicating how many characters to delete. A number reader for this command can also be used in connection with other commands. Other options conceivable include deleting all characters up to and including the next "end of line" character, thus enabling the editor to delete one or more lines.

Cursor Commands

Commands which position the cursor are fairly easy to implement, since they require no moving of text. Commands to increment and decrement the cursor, and to move it to the top and bottom of the buffer, are trivial to implement. The only thing to worry about is the possibility of moving the cursor outside the buffer.

Commands to move up and down a line of text are just a little trickier. Assuming that the end of a line is designated an "end of line" character such as the carriage return character, the next line command searches the text buffer, starting at the cursor, for a carriage return; it then positions the cursor immediately after the carriage return. The previous line command searches backwards from the cursor for the second carriage return it encounters (the first carriage return is at the end of the previous line), and positions the cursor at the character immediately after this second carriage return. Numeric arguments can be used to move up or down several lines at a time.

The most complex cursor positioning command is the search command. Its function is to search the buffer for a user specified sequence of characters, and then to position the cursor where the sequence is found. The search may be forward or backward from the cursor.

A buffer will be needed to hold the search string: the sequence of characters to be matched. If you implemented a buffer for input, you can use it again here. To conduct the search, scan the text buffer for the first character; then, having found a match for it, check if the immediately succeeding characters match the rest of the search string. If they don't, resume scanning for the first character. Continue until a match is found or the buffer runs out. If the search is unsuccessful, the cursor is probably best left where it was before the search started.

Input and Output Commands

There are two kinds of input and output that the editor needs to be able to do. One is communication with the terminal; the other is interaction with long term mass storage such as cassettes, tape cartridges or floppy disks.

Terminal input has been implicit in the discussion of all the commands. Terminal output has two specific functions: giving the user messages and displaying parts of the text buffer. Messages should include complaints about illegal commands, notifications of whether searches succeeded, warnings about attempts to go outside the buffer, indications of failure in transfers to or from mass storage, and so forth. Typeouts from the buffer can be handled in various ways, but what is important is that it be easy to look at the vicinity of the cursor. A minimal print or display command might just print the line which includes the cursor character. For greater flexibility, arguments to indicate a number of lines could be implemented. With two arguments, a number of lines before the cursor and a number afterward could be printed out; if both arguments are zero, just the line containing the cursor character would appear.

The other part of input and output is based on the need to store your text for later use. It is not very efficient to have to type in your whole program anew every time you want to update it; it is far more convenient to have a copy of the previous version on magnetic or paper tape, disk, or whatever medium is available to you, so that you can load the editor's text buffer, make the necessary changes, and output the new version for later sessions.

This is simple enough if the entire text file with which you are working will fit in the text buffer (given that you have the necessary interfacing and IO routines). But when there is too much text for the editor to handle at once, it has to be broken down into buffer sized pieces. To do this, you need the ability to alternate between reading one file and writing another. If you have cassette storage, this means you should have two recorders with motor control. The trick can probably be done with one recorder, but only with painful quantities of rewinding and positioning via the tape counter as the tape is moved between the two files.

A large text file should be broken up into pages. This is done by inserting the form feed character control L into the text at reasonable intervals. The text input routine

Command	Explanation
Ixxx↑C	Insert xxx, any sequence of characters except ↑C, into the text buffer.
Dnnn↑C	Delete nnn characters prior to the cursor.
Fnnn↑C	Move the cursor forward nnn characters.
Rnnn↑C	Move the cursor back nnn characters.
A	Move the cursor to the top of the buffer.
Z	Move the cursor to the bottom of the buffer.
Lnnn↑C	Move forward nnn lines.
Bnnn↑C	Move backward nnn lines.
Sxxx↑C	Search forward from the cursor for xxx.
Uxxx↑C	Search backward from the cursor for xxx.
Y	Yank a page of text into the buffer after first clearing the buffer.
W	Write a page of text from the buffer.
C	Close the output file or device.
Tmm,nn↑C	Type out mm lines before the cursor, the line containing the cursor, and nn lines after the cursor.

Table 1: A set of possible commands for a text editor. These commands are just suggestions for the type of arguments that may be used in an editor language. Note that the character ↑ C stands for control C. Control C cannot be encountered in the text to be edited since it is used as an indicator at the end of a command.

can then be designed to read the input file up to and including the first control L encountered, copy it into the buffer area, and then wait for another command before reading another page in. This method allows you to fix up one page at a time, outputting each page before getting another page from the input file.

Table 1 presents a possible command language for an editor which includes the features discussed here. If a command takes no literal or numeric arguments, it needs no terminator; the next character is a new command. Where there are arguments, the character control C terminates the command. Note that with this command language, control C itself cannot be inserted; this should not be a great loss, since control C is normally an end of text indicator.

These commands will provide you with a small, yet versatile editor. If you want to expand further, options such as scratch buffers, iteration, and macro-commands are possible; for an example of a powerful editing program, see Digital Equipment Corporation's TECO. At the other end, you can leave out numeric arguments, searching and even line positioning for the simplest editor possible. It all depends on what you can afford in time, effort and storage.■

What's New?

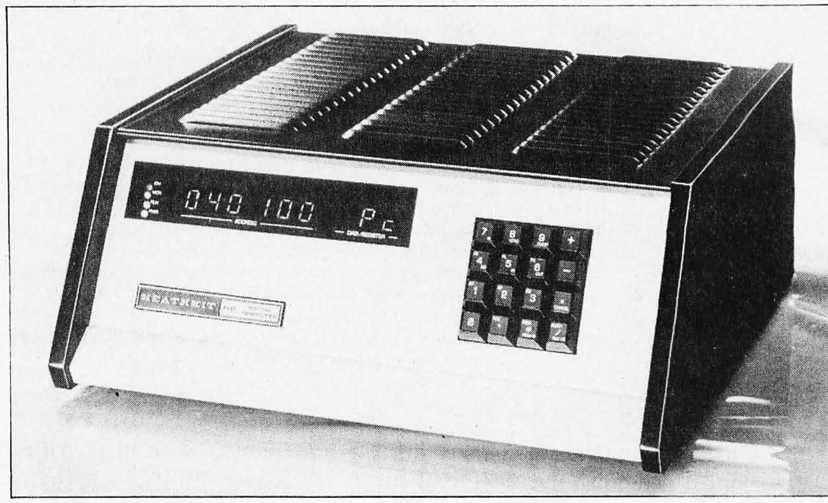


Photo 1.

The New Heathkit Computer Line

On June 1 1977 we traveled to the Heath Company's plant near Benton Harbor MI to take part in the "Heath Computer Product Press Party." This was a gathering of people from various publications to preview the new product line which Heath is introducing to the general public at the Personal Computing 77 show in Atlantic City NJ at the end of August. The major features of the new Heath product line are best described by pictures supplied to us at the press party, along with notes based both on the press release and our experiences.

The H8 Computer Kit (Photo 1)

The Heath product line has one 8 bit member, the H8 computer based upon the Intel 8080A design. This product, which sells at $375 for the basic processor box, includes cabinetry, front panel with 1 K ROM monitor, processor board and power supply. Memory cards are available with the H8-1 kit for 4 K bytes by 8 bits at $140 and the H8-3 expansion module adding another 4 K by 8 bits to the H8-1 for $95. (The cabinet shown is said to be capable of holding up to 32 K bytes in the form of H8-1 cards fully populated with memory chips.) Peripherals cards presently available include the H8-2 parallel interface card ($150) with three input and three output ports, and the H8-5 serial interface board ($110) which contains RS-232 interfaces at a number of data rates, 20 mA current loop interface, and an audio cassette interface which uses 1200 bps modified Manchester phase encoding at the "Kansas City" standard frequencies.

These systems come complete with software for immediate use, included as part of the price. At the press party, we exercised the H8 with Benton Harbor BASIC, a full implementation of Dartmouth BASIC for the 8080 which takes 8 K bytes and features dynamic storage allocation (programs expand and contract with use of data). The more complete 12 K Benton Harbor BASIC, extended, adds strings and other extensions to the 8 K version. Both versions have a number of user-oriented editing features which are both unique and useful as demonstrated to us at the press party. The design and coordination of the systems software for the H8 system were performed by Gordon Letwin, Heath's resident software wizard, whose title is Lead Systems Programmer for the Computer Products Line.

The H11 Computer "Kit" (Photo 2)

The second star in the Heathkit constellation is the top of the line H11 system, a confirmation of persistent rumors over the past year that Heath had allied itself with Digital Equipment Corporation. The H11 is an LSI-11 board from Digital Equipment Corp, mounted in a Heath supplied chassis and accompanied by Heath's switching power supply, backplane assemblies, and heavy duty documentation. It is intended for use with a terminal such as the Heath H9 or the LA-36 DECwriter which Heath will market, and at the time of initial deliveries the H11 is intended to be used as a stand alone paper tape system using the Heath H10 product as its paper tape facility. (Floppy disk systems and software are in the works but are not yet available from Heathkit.) The H11 computer itself is $1295 for the cabinet, LSI-11 board, Heath backplane and power supply. This price of course includes the built-in terminal support software of the LSI-11, the 4 K by 16 bit semiconductor memory of the LSI-11 board, and a terminal interface. Purchase of the Heathkit enrolls the user in the DECUS library and includes complete DEC systems software on paper tape: Editor, PAL-11 assembler, linker, ODT, BASIC and FOCAL languages. (BASIC requires additional memory.) Other components of the H11 line include the H11-1 4 K by 16 bit memory board at $275, the H11-2 parallel interface board at $95, and the H11-5 serial interface board at $95. The

Photo 2.

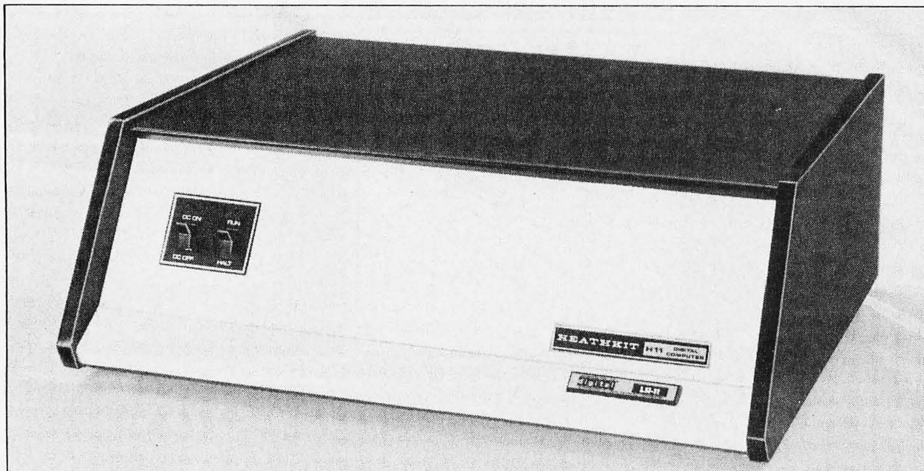




Photo 3.

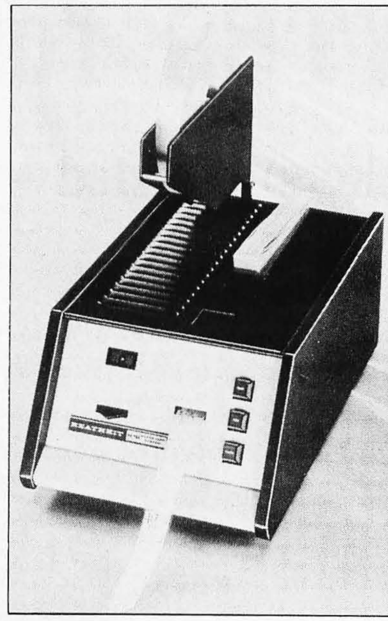


Photo 4.

H11 box will accommodate up to 20 K words of memory, according to the press release copy.

Other Systems Products

Both the H8 and the H11 are designed for use with other products which have generic applicability to any computer system. At present these include the H9 video terminal and the H10 paper tape reader and punch unit.

The H9 video terminal at $530 is probably one of the best buys available in a kit based terminal unit today. Its features include a full 67 key ASCII encoded keyboard, a 12 line by 80 character display format (upper case only), a formatting option for four columns of 12 lines of 20 characters built into its hardware, a block transmission feature, and a limited plotting mode of operation. The entire unit is contained in an attractive cabinet, as seen in photo 3.

The H10 paper tape reader punch is Heath's principal mass storage unit for use with the H11 product line as currently available (although, according to the people at Heath, this is by no means intended as the last word in that area). It can also of course be used with the H8, and any other computer which has a parallel 8 bit TTL compatible interface for input and output. Features of this unit, shown in photo 4, include a built-in power supply, stepper motor drive, a stand alone tape copy mode, and data rates of 50 characters per second reading, 10 characters per second punching. The price of the H10 kit is $350.

All Heathkit products can be purchased under Heath Company's revolving charge credit plan, a benefit which should be of major interest to potential computer hobbyists. Shipping dates for all new computer products are scheduled for the fall. For further information write for the "Computer Information Package" — Heath Company, Dept 360-26, Benton Harbor MI 49022.■

Circle 600 on inquiry card.

Computerized Morse Code Reception Package

Polaris Computer Systems has announced a Morse Code reception package for Altair (S-100) bus 8080 microcomputers. The package consists of a tone to DC converter module and complete software. The converter connects to the communications receiver via headphone jacks and to the computer via a parallel IO port. The converter contains a phase locked loop for tone decoding and adjustable center frequency and band width controls. Its design is highly immune to impulse noise. Provision for audio and visual synchronization (with an oscilloscope) of the incoming signal is provided.

The software adjusts for variations in transmission as each code element is received, allowing for manual or automatic transmission of CW at speeds ranging from 5 to 60 words per minute. Noise and dropout negating logic is included. Final output of received text is to an SIO port for display to a printer or CRT.

Package price in kit form for the converter, object program and complete documentation is $95. An assembled and tested version lists at $145. Source tapes and complete turnkey packages are also available. A schematic of the converter, object dump and documentation of the device are available for $17. Polaris Computer Systems is located at 3311 Richmond Av, Houston TX 77098.■

Circle 602 on inquiry card.

Note: *A group of other Altair (S-100) bus interface new products items begins on page 152.*■

XIMEDIA PRESENTS

The SOROC IQ120

CURSOR CONTROL. Forespace, backspace, up, down, new line, return, home, tab, PLUS ABSOLUTE CURSOR ADDRESSING.

TRANSMISSION MODES. Conversation (half and full Duplex) PLUS BLOCK MODE – transmit a page at a time.

FIELD PROTECTION. Any part of the display can be "protected" to prevent overtyping. Protected fields are displayed at reduced intensity.

EDITING. Clear screen, typeover, absolute cursor addressing, erase to end of page, erase to end of line, erase to end of field.

DISPLAY FORMAT. 24 lines by 80 characters (1,920 characters).

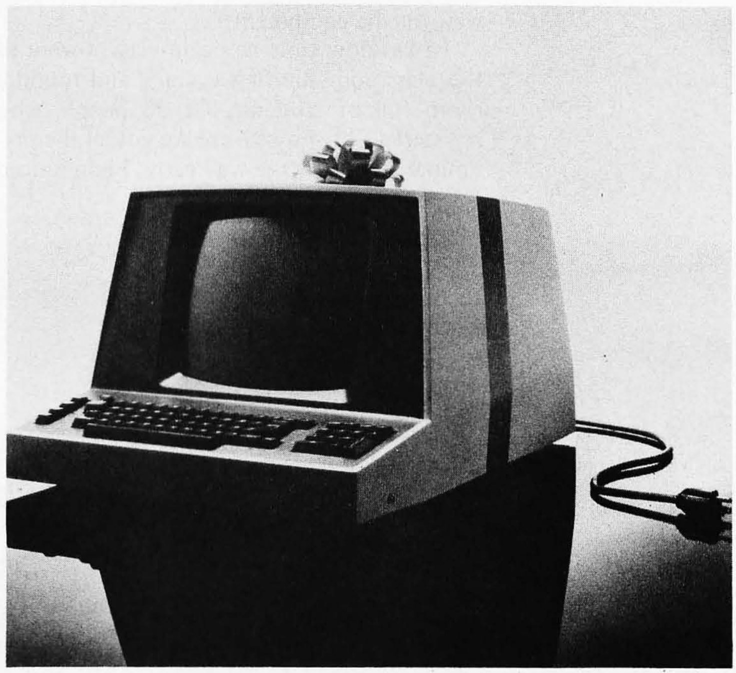
CHARACTER SET. 96 characters total. Upper and lower case ASCII.

KEYBOARD. 73 keys including numeric key pad.

REPEAT KEY. 15 cps repeat action.

DATA RATES. Thumbwheel selectable from 75 to 19,200 baud.

SCREEN. 12 inch rectangular CRT – P4 phosphor.



SPECIAL INTRODUCTORY PRICING

Kit $ 995.00 **Assembled $ 1,295.00**

(Price includes block mode, lower case and 24 line options.)

Specials of the Month

North Star **Micro Disk** with power supply and cabinet	Kit – $699
	Assembled – $799
IMSAI I-8080 with **TDL** ZPU	Kit – $825
	Assembled – $975
Digital Systems FDS Disk Drive with **CP/M** Software (assembled only)	Single – $1,750
	Dual – $2,350
Mountain Hardware **PROROM**	Kit – $145
	Assembled – $210
Vector Graphic 8K RAM	Kit – $235
	Assembled – $285

NOW WE'RE TOLL FREE
800-227-4440

(in California, Hawaii, and Alaska, call collect: 415/566-7472)

XIMEDIA OFFERS A FULL RANGE OF PRODUCTS FOR THE PERSONAL COMPUTER ENTHUSIAST AND THE SMALL SYSTEM DESIGNER. LET US QUOTE ON ALL YOUR HARDWARE AND SOFTWARE NEEDS.

OUR RETAIL STORE – THE COMPUTERIST™ – IS NOW OPEN IN SAN FRANCISCO. CALL US FOR DIRECTIONS.

XIMEDIA

1290 24th Avenue, San Francisco, CA 94122
(415) 566-7472

COD orders freight collect. Orders with payment shipped prepaid. California residents add 6% sales tax. Please allow 3 weeks for delivery.

Continued from page 28

matter of assembling the kits. The dean introduced me to two senior students who worked part-time in the school's electronics lab. These students, he said, would design a power supply and assemble the kits. The students began immediately on the power supply, and in about a week they finished the design. They gave me their parts lists, and I phoned the order in to the distributor. There was nothing left to do now except wait for the equipment.

It was now time to begin class. I went to the classroom the first evening and found a room full of students. Of 30 people who registered, 29 showed up. We got all the preliminaries out of the way early. I announced the class schedule, the type of equipment we would use, the grading scheme, my office hours, prerequisites, and all the other good stuff that's announced during the first class session. After some philosophizing about the history and potential of microprocessors, we got down to the business of the sequence. We started with the binary number system as planned. The lectures went very well. There was the usual supply of questions, answers, quizzes, homework, problems, and discussions.



Photo 2: A typical class session: designing, discussing, debugging and some time at the drawing board.

Meanwhile, the equipment was not coming in from the distributor as planned. We got the Evaluation Kits from Motorola, all except the read only memories. The discrete circuits just trickled in, a resistor here, a socket there, maybe a potentiometer or a voltage regulator. No amount of discussion, coaxing, cajoling, begging or screaming could get the components in. Some of the components, they admitted, had a 7 or 8 week lead time. Four weeks into the class schedule, we had approximately 90 percent of the pieces for one system. The students went to a local electronic shop to buy the missing items. This brought out our other major problem: namely, there were no terminals with which to talk to the system. I mentioned this to the dean, then borrowed an old CRT from North Electric for debugging purposes. At this point the dean ordered six new ASR-33s from Teletype. He explained that these would be used for other courses also.

One of the interesting developments of all this was that my student helpers, who were super careful in their work, discovered a few minor errors in the schematic that Motorola provided with their Evaluation Kit. The students routinely corrected the drawings and made up about 40 C size sheets for distribution to the rest of the students. In the process, I sent a copy of the revised schematic to the Motorola applications engineer in Cleveland to see if Motorola would object. He was quite surprised that the errors had not been found before. Not only did he not object, but he also was grateful for the sheet and asked if we could spare a few more!

After 11 weeks of class, including four weeks of excuses and apologies to the students, six lab stations were equipped. The packaging left a little to be desired (see photo 1), but the power supply and the kit worked, and the entire system served as an excellent educational vehicle. By this time, the lectures had covered the M6800 processor architecture, the Evaluation Kit system structure including the address space, and most of the M6800 instruction set. The students were more than ready to get their

Circle 127 on inquiry card.

hands on the system. Consequently, most of the next four weeks were spent in the lab, writing and testing small programs.

On the 15th week of class, 25 students took the final exam. The first trimester, which covered in significant detail a complete microcomputer system, was finally finished. In my opinion, it was a resounding success, even considering some of the problems.

After a week long vacation the second trimester began. In this trimester, applications and project work would be emphasized. Several lab projects were scheduled, some of which involved interfacing external hardware. Lecture time would be minimized, and most of what there was would feature guest lecturers and cover applications and related topics such as microprogramming and direct memory access techniques.

Unfortunately, only 15 students enrolled for the course. I attribute the sharp drop in part to graduation, but mostly to the fact that many students do not like to go to school in the summer. The students who did attend had an advantage though, because smaller class size gave them more opportunity to use the equipment.

The first part of the trimester was devoted to various number representations, conversion from one representation to another, and arithmetic operations in the representations. The first two lab exercises were the multiplication of two 8 bit numbers and the binary coded decimal addition or subtraction of decimal numbers up to four digits long. I scheduled these exercises first because my two technicians had graduated and nobody was available to assemble and test the hardware. I also thought that these exercises would accustom the students to writing fairly detailed programs. The students, however, had a difficult time with these programs. I found there were two major reasons for this.

The first was that there was no assembler available; all programs had to be written, assembled and entered into system memory by hand. This made for a lot of aggravating housekeeping and transcription errors. The second reason was that most of the students were electronic types and this was their first experience with programming.

Meanwhile, I decided that there would be three more lab exercises. The first would be storing data on a tape cassette using the Lancaster interface described on page 30 of March 1976 BYTE. The second would involve a row of 24 LEDs addressable as three memory locations. The third exercise could be analog to digital conversion using the circuit shown on page 70 of May 1976 BYTE.

By this time I had rounded up a technician, and he started to work about the fourth week of class. The cassette interface was finished first, but we never got it working right. It took about two weeks to figure out that coupling capacitors were needed on the TTL circuitry. Then we found that the tape was position sensitive, even after we changed the delays from 2.5 and 5 seconds to 5 and 10 seconds, respectively. In addition, the entire thing was assembled on a super strip, and they just don't lend themselves to reliable operation. The analog to digital conversion and lights projects went very smoothly. The hardware worked right just about as soon as it was built. The students had less difficulty with these two projects, and had more fun playing with them than they did with any of the other three. Photo 2 depicts a typical class session during the second trimester.

The second trimester went fairly smoothly, and passed without much incident. We had one guest lecturer, and we spent one evening at Ohio State University watching a demonstration of a Motorola EXORcisor. The only problems during the second trimester were a few minor glitches in some of the external hardware and the lack of software development facilities.

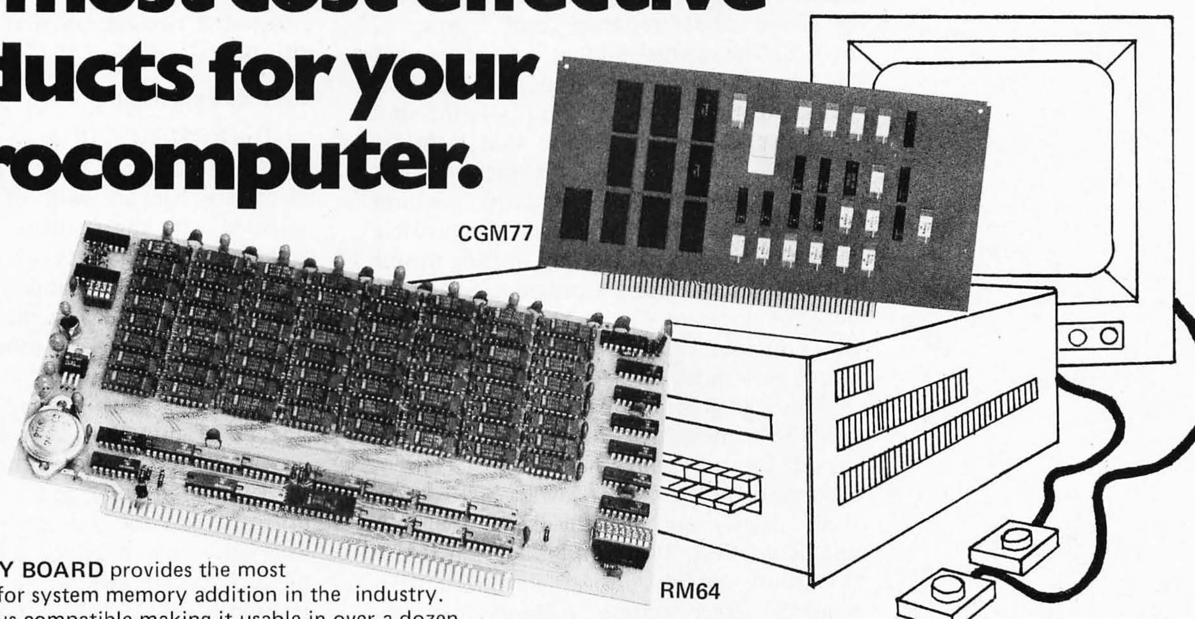
It was finally over. The 30 week excursion into microprocessors ended. For me, it was a great experience. Most of the hard work and sweat came in preparing the lectures and waiting almost endlessly for the hardware to arrive. The actual teaching of the course and conducting of the class were a pleasure. The students were perceptive, cooperative and eager.

The total cost to Franklin for both trimesters was about $1400. This includes six Motorola Evaluation Kits; discrete components, power supplies and extra programmable memory chips for nine kits; and a few miscellaneous components for the hardware labs. Not included in this figure is the cost of six brand new Teletypes.

I would like to express my gratitude to a few people whose contributions were invaluable. Dean Kelly provided his and Franklin's cooperation in the overall offering of the course. Mike Kabealo, the Motorola sales engineer, gave us the hardware which was the foundation of our lab. Dr Tony Lucido showed us his EXORcisor and its software, and he also provided us with an excellent example of how not to fix a Teletype. Finally, Lee Sensibaugh and Tom Richards, my two ace technicians, through their enthusiasm and tireless efforts, were the key factors in getting the laboratory together and working.■

Extensys Corporation.

The most cost effective products for your microcomputer.



THE RM64 MEMORY BOARD provides the most cost effective means for system memory addition in the industry. The board is S-100 bus compatible making it usable in over a dozen different microcomputer systems including ALTAIR and IMSAI.

It comes in three configurations: 32K, 48K or 64K bytes and is completely assembled, checked out and burned in for at least 50 hours prior to shipment. This complete testing procedure enables EXTENSYS to provide you a one year warranty on parts, labor and materials (assuming no misuse of the board occurs).

On board hardware is provided for:

- **Individual memory bank address selection in 8K increments;**
- **Complete dynamic refresh logic without loss of processing efficiency while programs are running;**
- **Board select logic which allows more than one 64K byte board per system;**
- **S-100 bus compatibility including on-board voltage regulator;**
- **Memory overlap which allows memory sharing the same address space to coexist in the same system; and**
- **Write protection in 16K blocks**

THE CGM77 COMPUTER GRAPHICS MACHINE is a fully tested system that provides the basis for developing the most sophisticated video graphics imaginable. The CGM77 board is S-100 bus compatible, making it usable on over a dozen different computer systems. The only additional components needed for a complete video graphics system are 8K bytes of RAM interfaced to the microcomputer system. The EXTENSYS RM64-32K RAM board is ideally suited for this.

Special control panels are available that provide universal control for interfacing with the graphics board. Output of the board is in the form of composite sync that can be used directly by a high resolution video monitor.

The heart of the system is the EXTENSYS GRAPHICS MACHINE. This is a sophisticated piece of software developed by EXTENSYS for the rapid implementation of video graphics. The GRAPHICS MACHINE permanently resides in PROM chips that are part of the board and therefore never need to be loaded. The GRAPHICS MACHINE software actually transforms the microprocessor into a much more powerful and sophisticated computer whose instruction set is ideally suited for the development of video graphics or video games.

Contact your local computer store or order directly from EXTENSYS

B–2

extensys corporation
592 Weddell Drive
Sunnyvale, California 94086

Please place my order for the following:

QTY	DESCRIPTION	AMOUNT
____	RM64-32K byte board @ $ 895 ea.	____
____	RM64-48K byte board @ $1195 ea.	____
____	RM64-64K byte board @ $1495 ea.	____
____	CGM77 computer graphics board @ $ 375 ea.	____
	Subtotal	____
	California residents add 6½% tax	
	TOTAL	____

Shipping and handling prepaid in continental United States

Please check method of payment:
__ Check Enclosed
__ BankAmericard No. ________ expiration date ________
__ Master Charge No. ________ expiration date ________

NAME ____________________
ADDRESS ____________________
CITY ________ STATE ______ ZIP ______ SIGNATURE: ____________
PHONE(INCLUDE AREA CODE) ____________

Circle 137 on inquiry card.

Continued from page 32

tape consists of a start bit, 8 data bits and a parity bit. Parity of each byte is automatically checked when a cassette program is loaded into memory. A light and warning tone tell the user immediately if a parity error occurs. This makes cassette volume and tone adjustments possible. Another light provides a visual indication of tape data. This lets the user manually position the tape when recording several programs on the same cassette. The last byte loaded into memory from a cassette is always displayed on the TV screen.

The graphic video display consists of an array of spots on the TV screen. There are 64 spots in the horizontal direction. The number of spots in the vertical direction can be varied, under program control, from 32 to 128. The normal display is 64 spots wide by 32 spots high. This array of 2048 spots represents 256 bytes. If a bit is 1, the spot is white; if a bit is 0, the spot is black. Changing the states of the memory bits with a program creates patterns, pictures or numbers on the TV screen. It is possible to create moving patterns by shifting the bit patterns around using a program. Switching between different display maps by changing the contents of a single address register allows very quick display changes.

This graphic display uses the RCA CDP1861 integrated circuit which was designed for use with the COSMAC microprocessor. It provides the lowest cost graphic video display currently available with any microcomputer. It is limited to fairly low resolution applications. Higher resolution requires larger refresh memory capacity and much more expensive circuits. In five years of experience I have found this resolution adequate for large numbers of applications. A very inexpensive color graphics add on is being planned for this display.

Physically, the COSMAC VIP consists of a single 8.5 by 11 inch (21.6 by 27.9 cm) printed circuit card that sits on rubber feet (see photo 1). The hexadecimal keyboard occupies the lower right hand corner of the card. The power pack plugs into the wall and is connected to the card with two wires. Three shielded cables are needed for connection to the video monitor, the cassette recorder mike input and the earphone output jack. These cables are readily available at local electronics outlets.

Assembly time for a COSMAC VIP kit is two to three hours. It involves soldering the components into the card. No individual wiring is required, other than the above external connections. The assembly instructions and printed circuit card were designed with the experienced kit builder in mind. If you have never built a kit you would be better off finding a friend with a sharp eye, steady hand and fine point soldering iron to help you out. Assembled versions will be available and a plastic cover for the card is planned. The printed circuit card can also be mounted in your own box with the keyboard, single operating switch and three LEDs remounted on the surface of the box. Only 19 integrated circuits are used in the 2 K byte standard system. A single +5 VDC supply comes with the kit and powers the computer. Only 350 mA, average current, is required with most of the current used by the NMOS programmable memory. The CDP1802 COSMAC microprocessor is the key to the low power and relatively few integrated circuits required for a system of this type.

System Expansion

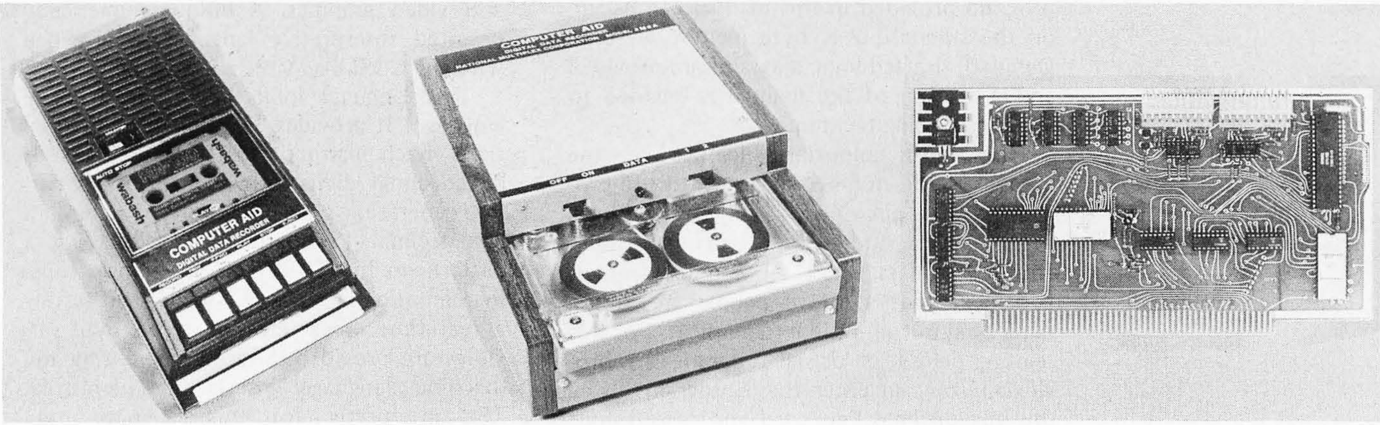
The COSMAC VIP was designed as a complete, stand alone, graphic home computer. Hardware hackers will, of course, not be satisfied with the standard system and will want to add a variety of gadgets of their own. Expansion capability was therefore designed into the system. First, you can increase memory to 4 K bytes by just adding four ICs to the printed circuit card. This permits more sophisticated programs for a system of this class. A comprehensive, 44 line interface is provided that can be used to add almost anything to the system including up to 32 K bytes of programmable memory. This interface provides all the COSMAC microprocessor signals. A fair amount of technical skill is, of course, required by hackers who actually use these lines.

For novices who want to add existing IO devices such as relays, music synthesizers, printers, or an ASCII keyboard, an easy to use parallel IO port option is provided. Adding four readily available ICs to the printed circuit card provides an 8 bit output port and an 8 bit input port plus handshaking signals on 22 interface pads. A standard 22 pin card socket can be used for connecting external devices to these ports. The output lines can drive two TTL loads. The input lines are high impedance, 22 K ohms or more.

Software and Documentation

Two manuals are provided with the COSMAC VIP system. The COSMAC microprocessor user manual is for those who want to get involved with the details of machine language programming. The COSMAC VIP manual provides all the information required to assemble, run, program and troubleshoot the system. Every

THE LATEST IN TAPE SYSTEMS



MODEL CC-8 - $175.00

4800 BAUD CASSETTE RECORDER

An ASYNCHRONOUS NRZ type Recorder with remote motor start/stop. Error rate 10^8 at 4800 BAUD. Can be used from 110 to 4800 BAUD into a UART or "Bit Banger PIA" – no clocking required. This is not an audio recorder. It takes RS232 or TTL signals from the terminal or computer and gives back the same signals. No audio interface is used. Motor start/stop is manual or through TTL or RS232 signals.

Tape speed is 3.2"/second nominal; 1.6"/sec. optional. 110 volt, 60 Hz, 5 watts. (220 Volts on special order). Can use high quality audio cassettes (Philips Type) or certified data cassettes. Can be used in remote locations from a 12 Volt battery.

Recommended for DATA LOGGING, WORD PROCESSING, COMPUTER PROGRAM RELOADING and DATA STORAGE. Especially recommended for 6800 systems, 6502 systems, 1800 systems and beginners with the 8080 systems. Manual control except for motor start/stop. 6800 or 8080 software for file or record searching available on request with order. Used by major computer manufacturers, Bell Telephone and U.S. Government for program reloading and field servicing. AVAILABILITY – Off the shelf.

MODEL 3M3A - $220.00

50 KILOBAUD CARTRIDGE RECORDER

This is a self clocking (1/1) high speed recorder. Loads BASIC in under 2.0 seconds. Recording is done at 19.2 Kilobaud. Playback at 30 Kilobaud. File or Record searching is done at 50 Kilobaud and loading is automatic. Worst case access time about 2 minutes for up to 2 megabytes on the 3M Data Cartridge.

Tape speed 10"/sec. on record, up to 30"/sec. on playback. Records one clock track and one data track on each pass (2 passes). Recording is NRZ unencoded from RS232 or TTL signals.

This recorder requires one Parallel port for motor control, and one serial port for data and clock. (Cannot be used with UART's or UART boards such as the 3P+S). Used with USART's, ACIA's or other 1/1 clocking I/O devices under software control only. No manual controls. Software for 8080 and 6800 available. Power supply is built in, 110V, 60 Hz. 220 V, 50 Hz for export.

AVAILABILITY – 90 Days

2SIO (R) CONTROLLER

$190.00 ($160.00 Kit)

PROVIDES MONITOR AND TAPE SOFTWARE in ROM TERMINAL and TAPE PORTS on SAME BOARD CONTROLS ONE or TWO TAPE UNITS (CC-8 or 3M3A)

This is a complete 8080, 8085, or Z80 system controller. It provides the terminal I/O (RS232, 20 mA, or TTL) and the data cartridge I/O, plus the motor controlling parallel I/O latches. Two kilobytes of on board ROM provide turn on and go control of your Altair or Imsai. NO MORE BOOTSTRAPPING. Loads and Dumps memory in hex on the terminal, formats tape cartridge files, has word processing and paper tape routines. Best of all, it has the search routines to locate files and records by means of six, five, and four letter strings. Just type in the file name and the recorder and software do the rest. Can be used in the BiSync (IBM), BiPhase (Phase encoded) or NRZ modes with suitable recorders and interfaces.

This is Revision 7 of this controller. This version features 2708 type EPROM's so that you can write your own software or relocate it as desired. One 2708 preprogrammed is supplied with the board. A socket is available for the second ROM allowing up to a full 2K of monitor programs.

Fits all S100 bus computers using 8080 or Z80 MPU's. Requires 2 MHz clock from bus. Cannot be use; with audio cassettes without an interface. Cassette or cartridge inputs are RS232 level.
AVAILABILITY – Off the shelf.

OVERSEAS: Export Version 220 volt 50 hz. Write factory or: Megatron-Datameg, 8011 Putzbrunn, Munchen, Germany; Nippon Automation 5-16-7 Shiba, Minato-Ku, Tokyo, Japan; Hobbydata, FACK 20012, Malmo, Sweden; G. Ashbee, 172 Ifield Road, London SW 10-9ag: Trintronics, Ltd., 186 Queen Street W., Toronto, Ontario, Canada; EBASA, Enrique Barges 17, Barcelona 14, Spain; ARIES, 7, rue Saint Phillipe du Roule, 75008 Paris; Microlem 20131, Milano, Italy; Eagle Electric, Capetown, S. Africa.

For U.P.S. delivery, add $3.00 Overseas and air shipments charges collect. N.J. Residents add 5% Sales Tax. WRITE or CALL for further information. Phone Orders on Master Charge and BankAmericard accepted.

Canadian Distributor:
Trintronics Limited
186 Queen Street West
Toronto, Canada M5V 1Z1
Tel: (416) 598-0262

NATIONAL MULTIPLEX CORPORATION

3474 Rand Avenue, South Plainfield NJ 07080 Box 288 Phone (201) 561-3600 TWX 710-997-9530

Circle 129 on inquiry card.

program provided in this manual can be run on the standard 2 K byte memory system. Detailed, drafted logic diagrams are provided and a section of the manual is devoted to troubleshooting techniques.

The most important feature of the manual is the number of ready to use program listings provided. I am one of those people who find it hard to understand anything unless I can see some examples. I suspect that a lot of people are in the same boat but don't like to admit it. For this reason, plenty of actual programs are provided. You can enter these programs using the hexadecimal keyboard and record your own cassette library. These programs were developed and debugged using the COSMAC VIP system itself to thoroughly test the operating system. Most of the programs were written in the interpretive language provided with the COSMAC VIP system.

Several test programs are provided to help you identify and analyze hardware problems should they occur. A short test program that can be loaded from the hexadecimal keyboard allows recording of a test cassette for analyzing tape problems. A special memory test program is provided that will detect most types of memory bit failures and will inform you which circuit to replace.

20 video game programs are provided. These games include target shooting, number guessing, picture or pattern generation, shooting stars, etc. Let's be honest, a primary application for home computers is games. There is nothing to be ashamed of if that's all you do with your computer. Chess, Go, bridge, tennis and baseball players do not constantly apologize for playing games. The sales of commercial games and the newer video games show that games fill a real need. Many people spend most of their spare time watching people getting paid to play games on TV. Games are fun, educational and satisfy basic human needs. The computer is a super toy. Why not relax and enjoy it? The COSMAC VIP game set permits the whole family to enjoy it with you. Starting with games may even develop latent programming interests that solving equations won't.

Designing Programs

This is where the frustration starts for many beginners. A number of high level languages have been developed to make computers easier to program. These are invariably English language oriented. This is nice in theory but uses up a lot of memory in practice. The more popular languages are also aimed at math and data processing applications rather than control, games and video graphics. A unique numerically oriented interpretive language is provided with the COSMAC VIP.

This language looks a lot like machine language. It provides 31 elementary instructions. Each instruction is two bytes or four hexadecimal digits. Single instructions permit generating a random byte, reading in a hexadecimal keyboard digit, displaying a pattern on the TV screen, sounding a tone, incrementing a variable, setting or testing a real time clock, etc. Sixteen 1 byte variables are provided. Subroutine nesting and machine language inserts are permitted. The interpreter for this language only occupies 512 bytes of programmable memory. Programs written in this language are extremely compact. A programmed video kaleidoscope requires less than 64 instructions. A rocket launching space ship intercept video game only requires 104 instructions including on screen decimal scoring. It is possible to play tic-tac-toe against the computer using a 235 instruction program that includes random mistakes by the computer. There is space for programs containing up to 592 instructions in the standard 2 K byte system.

I've been using this type of hexadecimal interpretive language programming for a number of years. It has been learned and used by children, engineers and programmers with no difficulty. An occasional professional programmer may grumble about the lack of self-documentation and the need for patches, but the money that these features cost isn't coming out of his pocket. Newcomers to computers might even find this COSMAC VIP language somewhat easier to understand than conventional high level languages. This is particularly true if they come from a hardware background. It is certainly cheaper to implement this type of interpretive language. Ease of understanding is difficult to measure since everyone is different. What's easy for one person may be difficult for another. Newcomers may welcome a programming approach that simplifies programming without straining their memory budgets. The home computer field is still young enough to tolerate experimenting with a variety of languages.

Some home computer users will want to get involved with machine language programming. They may be interested in developing their own languages or learning microprocessors at the detail level to enhance their careers. The COSMAC VIP can be directly programmed in machine code from the hexadecimal keyboard, and machine language programs can be saved on cassettes. The operating system permits examination of the

processor's registers to facilitate debugging machine code programs. The external interface permits substituting a user's read only memory based operating system for the standard one, if ever required.

Summary

COSMAC VIP is a complete computer on one printed circuit card. It is oriented toward control and video graphics. In price and performance it fits in between low priced machine language trainers and more expensive high level language oriented systems. It is ideally suited to personal use at home and educational use in schools.

COSMAC VIP is being made available by RCA. It is the result of five years of development effort in my laboratory which was aimed at producing a cost effective hobby computer. A low level interpretive programming language, test programs and 20 video game programs are provided with COSMAC VIP as it is coming to market for the first time. Assembled and covered up versions are planned. Add ons such as color graphics are being developed. Family fun, low cost, and flexability were primary design goals, which I believe we've accomplished with this product.■



Exciting lectures sponsored by

BYTE
The Small Systems Journal

Personal Computing Expo is endorsed by BYTE magazine, whose staff has contacted prominent speakers for an exciting series of lectures.
Visitors will be able to attend these meetings **free of charge.** The lectures will not conflict with each other eliminating the worrysome choice among several equally important topics. In addition, they will be repeated on the next day to give you a second chance if you missed a topic.
Lectures are typically 30 minutes, often with demonstrations and an additional 15 minutes for questions.

Portia Isaacson Ph.D. . . . Saturday 11 AM and Sunday 12 Noon

Co-owner of the Micro Store, a personal computer store, in Richardson Texas actively engaged for 12 years in the computing field in industry and at universities.
Member of the ACM and IEEE, and chairman of the 1977 National Computer Conference.
Author of many articles in professional journals and magazines.
Received a Ph.D. in Computer Science from the Southern Methodist University.

Personal Computing: An Idea Whose Time Has Come

A review of what has happened so far in the personal computing field, and an outlook into future developments, including those in the computer assisted home. Slide demonstration.

Sol Libes . . . Friday 6 PM and Saturday 10 AM

President of the Amateur Computer Group of New Jersey
Teacher of electronics and computer programming at a community college
Author of 10 books (working on the 11th) and several hundred magazine articles in electronics and computing
Received an award for "The Outstanding Amateur Computer Hobbyist of 1976" in Atlantic City by Personal Computing 1976 show, and BYTE.

How to Get Started

A discussion of typical home computer systems and their essential hardware and software components.

John H. Dilks III . . . Saturday 1 PM and Sunday 2 PM

President of Personal Computing Inc. and tradefair director of the Personal Computing shows in Atlantic City in 1976 and 1977.
Experience with various computer systems since 1962.
Employed by Western Electric Co. Inc., division of the Bell System.
Member of the Amateur Computer Group of New Jersey and of the Philadelphia Area Computer Society.
Teaches microcomputer courses in an adult evening program at a vocational school.

Innovative and Unusual Computer Applications for the Home.

Discussion of "far-out" applications of microcomputers and electronic technology for home use, such as a child locator and warning device, a home security system, etc. Slide demonstration.

Robert S. Jones . . . Friday 7 PM and Saturday 2 PM

Publisher of Interface Age Magazine
Prior experience in sales and marketing for the semiconductor industry, including Intel, National Semiconductor and Analog Devices Inc.

Personal Computing for the Business Man

Evaluating business applications for micro computers, including slides showing selected applications.

Louis E. Frenzel . . . Saturday 3 PM and Sunday 3 PM

Director of Computer Products at Heath Company, involved in the planning of new computer products.
Prior to Heath, with McGraw Hill in product planning and design of educational electronic kits.
Prior experience including computer engineering for eight years
Author of several books, home study courses and numerous magazine articles in electronics and computers.
Received a BS in electronics from the University of Houston and a MEd from the University of Maryland

How to Build Personal Computer Kits

Tips for successful kit construction. Benefits of kit products for the personal computer user. Including slides showing selected computer kits.

Carl Helmers . . . Saturday 4 PM and Sunday 4 PM

Editor-in-Chief and co-founder of BYTE magazine.
Obtained computing experience as a personal way to accomplish artistic and technological goals in music.
Graduated in 1970 with a BS in Physics from the University of Rochester, NY.
Worked for several years at Intermetrics, Inc. in Cambridge, Massachusetts on the NASA Space Shuttle Project.
Prior to working with BYTE, publication of a small computer newsletter on a part-time basis.

Computers and Music

How to create music with computers. Problems of performing electronic music, music under computer program control and computer music in conjunction with traditional instruments. Illustrations and examples from personal experience.

Jack L. Davies . . . Friday 8 PM and Saturday 5 PM

President of Pan Atlantic Computer Systems GmbH, a distributor of various micro computer systems in Europe.
Extensive experience in using minicomputers and microcomputers in the US Military Schools in Europe. Designed and developed numerous games and educational programs for students in these schools.

Microcomputers in Education

Discussion of the many possibilities of using microcomputer systems in schools. Effect of personal computers on students.

David Fylstra . . . Saturday 6 PM and Sunday 5 PM

Member of the research staff of the Telecommunications Sciences Center at Stanford Research Institute for more than two years. Specialized in microcomputer software and computer simulation of speech processing systems.
Graduated in 1974 with a BA in English and Psychology, Stanford University, Phi Beta Kappa.
Active in the research on communication systems and devices for the deaf.

Speech Analysis and Synthesis for the Amateur

Using the personal computer as a device to analyze the acoustical foundations of speech and to formulate rules for the control of the speech synthesizer.

Max Mathews Ph.D. . . . Friday 9 PM and Saturday 7 PM

Director of Acoustical and Behavioral Research, Bell Labs
Author of **The Technology of Computer Music,** and numerous articles.
Scientific Advisor to the Institute for Research and Coordination of Acoustics and Music (IRCAM)
Dr. Mathews is often regarded as the "Father of Computer Music"

Pure Digital and Real Time Music Synthesis

The use of the digital computer as a musical instrument with which composers and performers create and play music. Slide and tape demonstration.

Carl L. Holder . . . Saturday 8 AM and Sunday 6 PM

Director of Product Management, Planning and Communications at Information Terminals Corp. for five years.
Prior experience, including Memorex Corp., in the area of magnetic media development and testing.

Present and Future Storage Devices

Survey and discussion of current devices and media, including latest technological developments like the charge coupled devices and magnetic bubble memories. Costs, advantages and disadvantages of these devices for the personal computer user. Accompanied by slides.

DAILY TUTORIALS

There will be two tutorials offered each day, one aimed at those participants who have little or no experience with microprocessors, and the other for those already experienced with microprocessor systems.

BASIC COURSES

1. Development of Microcomputer Systems for Business Use

Sy Ratner, Citibank, N.A.
- system design concepts
- economic efficacy of microcomputers versus large centralized computer systems
- distributed processing definitions
- advantages and problem areas
- network design and architecture
- data communications aspects
- case study: design of a stand-alone workstation for data entry and retrieval

2. Development of Microcomputer Systems for Home Use

Cary Ringel, Con Edison
- survey of simple microcomputer control systems for home use
- selection of hardware: IC's, boards, kits, development systems
- programming and software aids
- interfacing: A/D and D/A conversion
- examples: design of a home control system; microcomputers for a music synthesizer; computer TV games.
- case study: use of the Motorola 6800 in design of a microcomputer system

3. Survey and Comparison of Microprocessors

Donald Lewis, Standard Microsystems Inc. and other speakers
- definitions and distinctions between ALU-chips, controllers, microprocessors, microcomputers
- current applications
- microprocessor architectures (bit-slice, 4-bit, 8-bit, 16-bit, minicomputer-type)
- vendor survey
- performance evaluation and criteria for selection

INTERMEDIATE/ADVANCED COURSES

4. Microprocessor Interfacing

Donald Lewis, Standard Microsystems Inc.
- interface components (peripheral interface chips, UARTS, etc.)
- interface standards (IEEE 488, RE 232C, S-100, etc.)
- interfacing to keyboards
- interfacing to casette and floppy disk drives
- interfacing to display devices
- case study: how to design a CRT terminal

5. Microprocessor Programming and Software

Donald Lewis, Standard Microsystems Inc.
- software design: flow-charting, setting breakpoints, documentation, etc.
- assembly language for the Intel 8080, 8085, Z-80, Motorola 6800
- instruction types and addressing techniques
- use of the stack
- interrupt handling and direct memory access (DMA)
- software development aids
- high level languages for microcomputers

6. Technology Analysis and Forecast of Future Microprocessor Structures

Daniel R. McGlynn, U.S. Philips Corp.
and Will Mathys, MOS Technology Inc.
- emergence of specialized computational elements (SCE)
- architectural evolution (stack processors, reconfigurable architectures, multi-level logic)
- resource management techniques
- software evolution (nano-programming, extensible instruction sets, structured programming modules, very-high-level languages)
- evolution of semiconductor technology of microprocessors
- microprocessor architecture at the chip level
- case studies: design of MOS Technology's new 8-bit and 16-bit processors

Personal Computing Expo to be produced by H.A. Bruno & Associates

H.A. Bruno & Associates, Inc., has been prominent in the exposition and promotion fields since 1923. Highly skilled in the production and promotion of consumer and trade shows, the company currently promotes the American Energy Expo, the National Boat Show, Auto Expo/New York. Promotion assistance also is currently rendered to the National Computer Conference and the Triennial IFIPS Congress in Toronto.

The show producer has promoted successful shows in the New York Coliseum every year since the building opened in 1957. Staff personnel are thoroughly familiar with the building, its services, management and labor.

Interesting, educational exhibits of Personal Micro Computers

The lectures and tutorials are not just theory! You will be able to see a multitude of micro computers. Try out various demonstrations: fascinating games, even in color; small business applications; computer produced speech; music; color graphics; and many more. Micro computers are not only small and portable, also you will need only a "micro budget" to be able to take them home.

Show Hours and Admission

Personal Computing Expo hours are as follows:
Friday, Oct. 28 — Noon to 10 p.m.
Sat. Oct. 29 — 10 a.m. to 10 p.m.
Sunday, Oct. 30 — Noon to 7 p.m.
General Admission: $5.00 (includes free BYTE lectures) per day.
Two-day Tickets: $9.00 (advance sale only)
Three-day tickets: $13.00 (advance sale only)

General Information

You may find it advantageous to purchase two or three-day admission tickets in advance. These are available by mail only, no later than October 10, 1977. Use coupon below.

Group rates (10 or more persons) qualify for $1.00 off regular prices. Arrangements must be made by mail prior to October 10, 1977.

Special arrangements have been made if you desire to stay overnight. Our headquarters hotel, the Barbizon-Plaza, is located on Central Park South, two blocks from Columbus Circle. Single rooms available at $34.00 per night; $40.00 double, plus tax. There's a weekend plan: $22.95 daily, plus tax per person, double occupancy . . . includes breakfast (brunch on Sunday) and meal gratuities. Children under 14 in same room with parents, free.

For hotel reservations and information, call toll free (800) 223-5493. From New York State call (800) 223-5963.

For those traveling to New York by air, American Airlines offers a convenient service through arrangement with Personal Computing Expo. For information, call toll free (800) 433-1790. In Texas the number is (800) 792-1150. From the West Coast, round trip fare via American is only $227.00.

20,000 persons are expected to attend and view the more than 200 exhibits by personal computer manufacturers and retailers.

Personal Computing Expo will occupy the 4th floor of the New York Coliseum. It is located on 59th Street and Columbus Circle — the geographical center of New York City. Garage parking in the building is available.

For answers to any questions pertaining to your attendance at Personal Computing Expo, contact the Show Manager, Ralph Ianuzzi, at Area Code 212/753-4920.

Advance sale of tickets available . . . Three days $13 Two days $9 . . . General Admission $5 MAIL THIS CONVENIENT ORDER FORM NOW!

Personal Computing Expo admission is $5.00 per day. Advance reservation eliminates waiting in line. Order advance tickets with this coupon. Admission ticket includes access to exhibits, lectures and tutorials.

Please send me ______________ advance registration tickets for three days, October 28-29-30. Total cost $13.00 per person.
Please send me ______________ advance tickets for two days, October ______ and October ______. Cost is $9.00 per person.
Please send me ______________ advance tickets for one day, October ______. Cost is $5.00 per person.

Make all checks payable to PERSONAL COMPUTING EXPO, and mail to: Personal Computing Expo, 78 East 56th Street, New York, N.Y. 10022.

Name ______________________ Amount enclosed $ __________
Address ______________________
City ______________ State __________ Zip __________

Book Reviews

Expanding Your Computer Library

What do you do with your system? Where is all the applications software? These are questions frequently posed by people who are trying to find out about small computers. Aside from games, which are a challenge to write as well as play, applications software seems somewhat scarce. In this article, I recommend some books that can introduce you to the world of computer applications. Most of the books discussed below contain program listings in BASIC or FORTRAN. Others do not, but are useful for the wealth of ideas or methods presented.

At the head of my list is a remarkable book, *Scientific and Engineering Problem-Solving with the Computer*, by William Ralph Bennett Jr (Prentice-Hall, NJ, 1976, hardcover, $17.95). Based on a computer course which Dr Bennett taught at Yale, the book introduces you to BASIC programming, then rapidly branches out to explore many unusual, but highly fascinating problems in various areas. Chapter headings include "Plotting and Graphic Display," "Dynamics," "Random Processes," "Wave Motion and Fourier Series," "Electronics and Communication," and "Language" (a singularly intriguing subject; ciphers, random text generation, and computer identification of authors are some chapter subheadings). Program examples introduce you to computer applications in these areas, and there are do-it-yourself problems to solve. References are given at the end of each chapter to facilitate further research. A reasonable knowledge of mathematics, such as college level calculus, is useful, but certainly not necessary to enjoy this text. Dr Bennett's writing style is refreshingly informal, and quite humorous in places. A word of warning: You will be extremely frustrated if you buy this text and do not have a machine on which to experiment.

Numerical methods are the mainstay of programming. Unlike analytical methods, which yield answers in terms of symbols or equations (the way you did things in algebra, trigonometry, etc), numerical methods are computer oriented, and yield digital answers. Typical numerical algorithms (a prescribed step-by-step way of carrying out calculations) are used to approximate functions (trig, log, etc), solve equations, find areas and volumes (integration), or the inverse, rates of change (differentiation), solve differential equations, manipulate matrices, and numerous other procedures.

Though Bennett describes some numerical methods, there are more detailed books on the subject. Two of the best I have encountered are *Numerical Methods for Scientists and Engineers*, by Richard W Hamming (McGraw-Hill, New York, 1962, hardcover, $18.50) and *Numerical Methods for Digital Computers, Volumes 1 and 2*, by Anthony Ralston and Herbert S Wilf (Wiley, New York, Volume 1, 1960; Volume 2, 1967, $17.25 and $19.25 respectively). Both of these cover algorithms for numerical procedures mentioned above, and considerably more. Hamming's book is a text, with descriptive chapters on each major subject. There are no program listings, but the algorithms are not terribly difficult to translate into a computer acceptable form. The book can be used for self-instruction, and a knowledge of calculus is helpful. Ralston and Wilf's books are more reference-oriented. It is easy to find the algorithm for your particular need. The first (earlier) volume does not contain specific program listings, though flowcharts are abundant. The second volume does have listings, mostly in FORTRAN and ALGOL. This volume also includes an introduction to programming in both FORTRAN and ALGOL, though it is somewhat out of date. Ralston and Wilf's volumes are more useful once you have some experience with numerical programming.

A very fine text in numerical methods is *Applied Numerical Methods*, by Brice Carnahan, H A Luther and James O Wilkes

A Note About Prices in This and Other Book Reviews

Readers should be aware that publisher's prices are subject to change without notice. While we check the accuracy of prices at the time book review copy is prepared, occasionally prices may have changed by the time subscribers receive their BYTEs.

(Wiley, New York, 1969, hardcover, $21.95). This book uses solved and unsolved problems to illustrate the major concepts of numerical problem solving. Numerous useful program listings are given in FORTRAN, along with sample computer runs. Like Hamming's book, this is a text, but is less general in scope. The programs, however, make it most attractive, as it is sometimes easier to understand an algorithm when presented in coded form. The problems chosen as illustrations are engineering oriented. For example, the book uses radiant interchange between plates, displacement of a cam-follower, flow in a pipe network, member forces in a plane truss, and transient behavior in a resonant circuit. The programs are organized as subroutines which you can easily use for other purposes.

Graphical presentation of data is discussed in Bennett and Carnahan et al, with some program examples. For those of you with some sort of computer graphics facility, or an interest in graphics, *Mathematical Elements for Computer Graphics*, by David F Rogers and J Alan Adams (McGraw-Hill, New York, 1976, paperback, $9.95) provides an excellent software "how-to" (unless you have a system with hardware graphics functions). The book has chapters for each major graphic function: rotate, translate, reflect, project, perspective and stereo view. A major bonus is a complete set of program listings for use in any BASIC that has true subroutine call capability (like FORTRAN). They can easily be rewritten to accommodate other dialects of BASIC. The book is an excellent reference and text. Some knowledge of analytic geometry and trigonometry is helpful.

While Carnahan et al briefly examine chemical engineering, a more thorough book on programming in chemistry is *FORTRAN IV in Chemistry*, by G Beech (Wiley, New York, 1975, hardcover, $23.50). In addition to programs for simulation of lab problems on the college level, there is much material on theoretical chemistry (quantum and classical mechanics), as well as practical lab problems. These problems are complemented by FORTRAN listings and sample runs. This is a specific text, and if your interest is in chemistry, you might easily find a computer application based on the material in this book. Topics include: numerical methods, kinetic and thermal problems, spectrometry, atomic and molecular orbital theory, and data acquisition in chemistry. Familiarity with the noncomputer approaches to chemistry problems and a knowledge of FORTRAN are useful in reading this book.

Bennett, Carnahan et al, and Beech have chapters devoted to statistics, an area in which the computer has proved of inestimable value. Market analysis, reliability testing, election results projections, and even scholastic testing would be nearly impossible without the computer. The basic programs are relatively easy to write, but the more advanced techniques, including analysis of variance and factor analysis, are considerably more complex.

Two books serve as excellent guides to statistics by computer. *Biometry, the Principles & Practice of Statistics in Biological Research*, by Robert R Sokal and F James Rohlf (W H Freeman, 1969, hardcover, $18.50) is a textbook of statistical methodology. It is well written and clear, and as an added feature includes a complete library of statistics programs in FORTRAN. These are all subroutines which you can assemble to perform the task at hand. The book is oriented to biological research, but the methods are easily adapted to other situations.

FORTRAN Programming for the Behavioral Sciences, by Donald J Veldman (Holt, Rinehart and Winston, New York, 1967, paperback, $5.98) is not as thorough as Sokal and Rohlf, but is more concise. It has a chapter devoted to learning FORTRAN, and a set of statistics programs in FORTRAN. These two books assume a knowledge of mathematics through elementary calculus, but the mathematical difficulty varies with the particular topic. These texts have served me well, and I have even been able to earn some extra income by doing statistical analysis on research data. Believe it or not, there is a large number of people who still do statistical calculations on nonprogrammable calculators.

A book related to my field of endeavor is *Computer Techniques in Biomedicine and Medicine*, edited by Enoch Haga (Auerbach, Philadelphia, 1973, hardcover, $24.95). This collection of articles is divided into two major parts: simulation and modeling, and data acquisition and management. The articles vary in complexity and specificity. Topics include simulation with applications in cardiovascular and pharmacologic research, data base accumulation, image processing, and the role and potential of minicomputers in medicine. Some topics contain sample program listings and flow diagrams. Other topics, especially those on data base management, are quite general, but applications would be highly dependent on the particular data base. The book serves as a good starting point for exploring the rapidly growing field of biomedical computing.

This has been a brief introduction to selected books on applications programming. Because my involvement in computing is oriented to scientific applications, the books I've covered here are mostly science and engineering tomes. The amount of FORTRAN usage in these books is also large, but it has been my personal experience that FORTRAN and BASIC can be directly translated, if attention is paid to proper language rules. An elementary text on FORTRAN is always handy to have as a reference for syntax. If you know BASIC, FORTRAN is not at all difficult to learn.

The books mentioned are not inexpensive, but they serve as good investments if they help you solve a problem, or pose a problem you become interested in solving. I would encourage you to examine these books, if you can, before you buy them. A well stocked college science or engineering library should have many of them available. This review might also serve to encourage you to write about your favorite computer books.■

Steven C Horii MD
40 Waterside Plaza, 15F
New York NY 10010

Continued from page 17

ation returns the result *15*. This dual use of a symbol, like the concept of valence applied to functions, may seem strange and even confusing upon first encounter. Familiarity soon reveals that adoption of this procedure is a source of considerable economy in the language. There are only a few operators in APL so far, but they impart great power to the language. Implementation of new operators will likely highlight subsequent development of this exciting language.

Some insight into the evolution of APL has been provided by the innovator of that language, Kenneth Iverson, and his collaborators. When the history of its development is widely appreciated, APL will be seen as one of the great intellectual achievements of this century. Iverson is a mathematician who is keenly aware of inconsistencies that have long plagued his discipline and is eager to avoid those blemishes in APL. Nevertheless, the distinction drawn here between function and operator was not made in the earliest versions of the language. Only after operators were recognized as "composite functions" did the matter receive the attention that brought it into line with the generality that is a basic theme of the language.

Arnold certainly is not alone in using the older terminology. He cites a widely used textbook that deals rather lamely with this point. A more recent textbook on the language (one author a mathematician and prominent user of APL) uses the terms function and operator interchangeably and appears determined to promote the confusion that originators of the language have been at pains to dispel.

BIBLIOGRAPHY

1. Falkoff, A D and K E Iverson, "The Design of APL," *IBM J Res Develop* 17:324-334, 1973.

2. Forsythe, G, "What to Do Until the Computer Scientist Comes," *Amer Math Monthly* 75:454-462, 1968.

3. Iverson, K E, "APL in Exposition," *Tech Rept 320-3010 IBM Philadelphia Scientific Center,* 1972. A section of this excellent little volume entitled "The Computer: A Device for the Automatic Execution of Algorithms" is particularly appropriate to your journal and should be reprinted if at all possible.

An anonymous contributor to *Scientific Research* (1969) posed two interesting questions: "To what extent are scientists beginning to use programming languages to describe or model their problems in preference to normal mathematics?" "Is it possible that researchers will some day automatically *think* in FORTRAN or BASIC instead of algebra or calculus?" Positive answers to those questions could scarcely have been framed then, even by enthusiastic users of those languages. If those enquiries now be made of APL, there can be little doubt of affirmative answers.

As Forsythe (1968) pointed out, "The most valuable acquisitions in a scientific or technical education are the general-purpose mental tools that remain serviceable for a lifetime." APL certainly fits more comfortably into that category than does any other language by which we now try to instruct computers. APL is sometimes referred to as Iverson's notation; hence it is appropriate to recall a remark attributed to Bertrand Russell: "A good notation has subtlety and suggestiveness which at times make it seem almost like a live teacher." That notion seems to have been crystallized in APL. Iverson will eventually be recognized as the teacher of the century.

The story of APL includes a lesson that our overstressed society might do well to ponder, namely the importance of providing some shelter from expediency for the development of any significantly new idea. The small team initially involved in design and implementation of APL were not subject to strong duress to get their kite in the air. They did not have to settle for the compromises by which committees have produced such regrettable languages as say PL/I. Instead, they were able to indulge the luxury of Quaker consensus in reaching important decisions. Our greatest debt to earlier computer languages may well prove to be the lebensraum they provided for the development of APL. This remarkable language will undoubtedly undergo considerable further evolution, some of which will surely be influenced by its use with microcomputers. It must be implemented throughout the computer world. Those involved with its development and implementation should consider its history very carefully.

Our thanks to Mr Anthony for his comments about Mark Arnold's article and about the present and future status of APL.■



FREE YOURSELF FROM THE ONE BYTE WORLD. MOVE UP TO THE TWO BYTE **TEXAS INSTRUMENTS TMS-9900 16-BIT MICROPROCESSOR** — WITH OUR — **"SUPER STARTER SYSTEM"** — TEC-9900-SS. SHOWN ABOVE, FEATURES HARDWARE MULTIPLY AND DIVIDE, 69 MINI-COMPUTER INSTRUCTIONS, 7 ADDRESSING MODES, EXPANDABLE TO A FULL 65K BYTES; MONITOR, TMS 9900 CPU, RAM, P-ROM, E-PROM, PROGRAMMER ALL ON ONE P-C BOARD BASIC OPERATING SYSTEM AS LOW AS **$299 UNASSEMBLED $399 ASSEMBLED AND TESTED** EXPLICIT **MANUAL** INCLUDED OR AVAILABLE SEPARATELY AT **$35,** TO LEARN MORE . . .JUST TEAR OFF A PIECE OF THIS AD, PIN TO YOUR LETTERHEAD & RETURN TO TECHNICO OR **CALL OUR HOTLINE 1-800/638-2893** TO RECEIVE FREE INFO-PACKAGE. --- DESIGN & TECH SUPPORT BY ROSSE CORP.

Circle 113 on inquiry card.

TECHNICO
INCORPORATED

9130 Red Branch Rd.
Columbia, Md. 21045
301-596-4100

THE TECHNOLOGY LEADER IN ELECTRONICS DISTRIBUTION

Why People Get Hooked on APL

Allen Atwood III
Rm 2-501
472 Memorial Dr
Cambridge MA 02139

APL, A Programming Language, is one of the most powerful interactive languages in use today. Its rich set of operators is oriented toward operating on groups of data with the same freedom that other languages operate on single data items. These operators give the language the breadth to handle an amazingly wide range of computing problems. This article explores APL's applicability to personal computing, and looks at some of the ways that APL can be practically implemented on the small system. Particular attention is given to the problem of storage management. References in the article to the internal workings of an APL interpreter for a microcomputer pertain to the APL interpreter currently under development by the Microsoft Company for use with the Intel 8080 chip. This interpreter (being implemented as this is written) will include a significant subset of APL, designed to contain most of the features available on large APL systems, except for those parts of the language which pertain to a multiuser environment.

The first question one must ask when thinking about the implementation of a new language is whether or not the language is capable of solving the problems commonly solved on that machine.

The first question one must ask when thinking about the implementation of a new language on a machine is whether or not the language is capable of solving the problems commonly solved on that machine. With APL, the answer is a qualified yes. Since the language is interpretive, it will never be as fast or as efficient as assembly language or a true compiler; one would not want to refresh a display from memory using APL. On the other hand, APL is a far less cumbersome programming language than assembly language. Clearly then, APL can be used to solve the problems (and play the games) that are presently implemented with BASIC, an area where APL excels. As an interactive system, like BASIC, APL is easy to program with and debug, and it is also considerably faster than BASIC.

The heart of APL's speed lies in the ability to express highly complicated algorithms with a few keystrokes. This gives the language a double advantage over BASIC. Part of the speed improvement will come about simply because APL programs are shorter than their BASIC counterparts. The interpreter wastes less time analyzing the program and spends more time manipulating the data. The other, more subtle, speed advantage lies in the speed of the functional subroutines that are built directly into APL. As an example, consider this BASIC routine which calculates the average of the data items in the N element array A:

```
1000  S=0
1010  FOR I=1 TO N
1020  S=S+A(I)
1030  NEXT I
1040  S=S/N
```

The routine exits with S being equal to the average of the N items in the array. Now compare that with an APL routine that performs the same function:

```
S←(+/A)÷⍴A
```

Notice that the entire loop, lines 1010 thru 1030, is reduced to the expression +/. Thus BASIC, while summing up the array elements, must go through great convolutions to achieve the desired result, including performing the subscript calculation A(I) N times and storing the temporaries S and I in memory each pass through the loop. This involves considerable overhead, including looking up S in the symbol table every time through the loop, skipping over any remark statements left inside the loop, calling the formula evaluator to analyze the LET statement in line 1020 each time through the loop, etc. The BASIC routine is also fairly cumbersome to use as a subroutine, because one must be sure that the loop variable I is not confused with some important piece of data (with the same name) in the calling program.

APL, on the other hand, does the entire calculation in a very fast loop internal to the interpreter, keeping the sum in a special memory location (or possibly an 8080 register, depending on the magnitude of the temporary). Not counting the time spent lexically analyzing the line, one would be hard pressed to write an assembly language program that performs the summation faster than APL does. The time overhead of analyzing the line, while measurable, is a fairly insignificant percentage of the total time spent, both because it is only done once, and because the line is only nine characters long. APL also relieves the programmer of the possible variable conflict between the loop variable I, and another use of I in some other part of the program, since it does not use a named variable as a temporary.

APL's operators can be used in combination to perform considerably less trivial tasks than averaging an array. Consider the following APL function called SORT. This function sorts an array of alphanumeric strings into an order specified by another string called an "alphabet" string. SORT takes two arguments: the string array to be sorted, and the alphabet to be used during the sort. As an example, we first assign the string ABCDEFGHIJKLMNOPQRSTUVW XYZ to the variable ALPHABET.

The request:

```
ALPHABET SORT FOO
```

would then cause APL to type the string array FOO sorted into normal alphabetical order. Other common permutations are easy to achieve; one could use the reverse operator (⌽) to get reverse alphabetical order. In this case, the proper APL incantation would be:

```
(⌽ ALPHABET) SORT FOO
```

Well, what does it take to write this magical SORT in APL? The function sort is defined as follows:

```
∇R←ALPHA SORT LIST;TEMP

[1] TEMP←(ρLIST←(ALPHA←''
     ,ALPHA)⍳LIST) [2]
[2] LIST←LIST[⍋LIST[;TEMP];]
[3] →2×⍳0≠TEMP←TEMP-1
[4] R←ALPHA [LIST]∇
```

In the first line (often called the function header) the function is declared as having two arguments which are assigned the values ALPHA and LIST during the execution of the function. TEMP is declared as a temporary variable, used only inside the function, and R is declared as the value variable of the function. Thus when the function is exited the value returned will be the value of R. All four of these variables are local to the function; that is, they will be cleared. Thus any space they use will be reclaimed upon exiting the function. Also, if there were any variables with these names already in existence, the old values are pushed onto the 8080 stack to be restored on exit from the function. (Actually, a pointer to the old value is pushed onto the stack, not the value itself.) This guarantees that there will be no conflicts between the function and the environment from which it is called.

Consider the deepest level of nested parentheses in line 1: The variable ALPHA has a string composed of one space catenated in front of it. This new string is first reassigned to ALPHA, and then used as the left argument of dyadic iota, with the right argument being the array (LIST) that we want to sort. This function returns a numeric array having the same number of rows and columns as LIST, and whose every character is replaced by the index of the matching character in ALPHA. A space would thus become a 1, because the first character of ALPHA is a space. This new numeric array is assigned to LIST, and then the shape function ρ (the greek letter rho), is applied to it. This returns a two element vector whose first element is the number of rows in LIST, and whose second element is the number of columns. The second element of the vector is selected by the indexing brackets, and is assigned to TEMP. TEMP thus points to the last column of LIST.

Lines 2 and 3 perform the actual sort. In line 2, the entire column pointed to by TEMP is extracted by the expression LIST [; TEMP], and sorted to produce a vector of indices used to index LIST again. This has the net effect of sorting LIST based upon the column pointed to by TEMP. In line 3, TEMP is decremented and its value tested. If TEMP is nonzero (indicating that the array has not been sorted on every column), control is transferred back to line 3 for another pass. If TEMP is equal to zero, then the interpreter proceeds to line 4 where LIST is used to index ALPHA and produce a character array sorted in the proper order. When the end of the function is reached, the old values of ALPHA, LIST, TEMP and R are restored, and the last value assigned to the sorted alphanumeric array R is returned.

Because the bulk of the sort is done by the grade up and index bracket operators, the algorithm is fairly fast. It is also quite short, being less than 120 characters. When compared with a BASIC routine that performs a similar function (see listing 1), the APL routine seems quite short indeed. The

APL tends to use large quantities of memory, reflecting its IBM/360 origins. The cause of this weakness is unfortunately also the cause of its strength: its ability to operate on groups of data.

```
100    FOR X1=1 TO N-1
       FOR X2=X1+1 TO N
       X=1
200    IF INSTR(A$,MID$(X$(X1),X,1))=INSTR(A$,MID$(X$(X2),X,1))
       THEN IF X=LEN(X$(X1))
          THEN NEXT X2,X1
          ELSE X=X+1
          GOTO 200
       ELSE IF INSTR(A$,MID$(X$(X1),X,1))>INSTR(A$,MID$(X$(X2),X,1))
          THEN SWAP X$(X1),X$(X2)
          NEXT X2,X1
          ELSE NEXT X2,X1
```

Listing 1: A BASIC string sort. This routine is designed to take a string of characters and arrange them in alphabetical order. Compared with the corresponding APL string sort (see article) this routine will typically take longer to execute and is more cumbersome to write. One of the drawbacks of APL, however, is that it invariably takes more memory space than other high level language interpreters.

The heart of APL's speed lies in its ability to express highly complicated algorithms with a few keystrokes.

BASIC sort, incidentally, is written in Altair extended BASIC; in standard BASIC it would be longer yet.

This routine also demonstrates one of APL's losing features: its memory inefficiency. APL tends to use memory in large quantities, reflecting its origins on the IBM/360. The cause of this weakness is unfortunately also the source of APL's strength: its ability to operate on groups of data. Instead of having single numeric temporaries inside a formula, as BASIC might, APL often designates entire arrays as temporaries. (Although BASIC can generate large character temporaries during string operations, they are limited to 255 bytes in implementations such as Microsoft's Altair BASIC.) These temporaries can consume significant amounts of memory, especially if they are numeric. APL requires at least 16 decimal digits of precision in its numbers, which in turn require eight bytes of storage per number.

The problem is aggravated by the size of the interpreter itself, which we expect will occupy around 24 K bytes if all the features of the language are implemented. Clearly, if APL is to become popular as a language among personal computer users, some way must be found to implement a practical APL interpreter on a small system.

The simplest way to solve the problem is to streamline the language, eliminating features until the interpreter will take up no more than 8 K bytes or so. One could limit the size of floating point data to four bytes rather than eight. Limiting arrays to one dimension, and thus eliminating most of the operators designed to operate on multidimentional arrays, would be a big saving. Eliminating the composite operators (except for the most important one, reduction) would save a lot of storage in the interpreter.

Once this point is reached, it would probably be better to forget about APL entirely and go back to BASIC, because all that remains is a simple desk calculator style language with a strange character set and a peculiar order of evaluation. The heart of the language is its ability to manipulate arrays, and when this feature disappears the language loses much of its usefulness.

It might seem more reasonable to eliminate only those operators, like matrix inner and outer products, which appear to be useful only to mathematicians; but as one becomes more familiar with APL, these functions become increasingly more useful in everyday applications. Matrix inner product, for example, is one of the most powerful tools in the language for extracting information from a stored array. Using matrix inner product, the entire APL sorting function can be reduced to only one line of code:

```
     ∇R←ALPHA SORT1 LIST
[1] R←LIST [⍋((' ',ALPHA)
    ⍳LIST)+.×⌽((⍴ALPHA)+1)*¯1
    +⍳(⍴LIST) [2] ; ]
```

This algorithm (from Gillman and Rose, *APL 360: An Interactive Approach*, John Wiley and Sons, New York, 1970) performs only one numeric sort, as opposed to SORT, which performs a numeric sort for each column of the array. Although this code is fairly obscure (especially to a person used to BASIC or FORTRAN), one becomes more comfortable with this sort of coding through time.

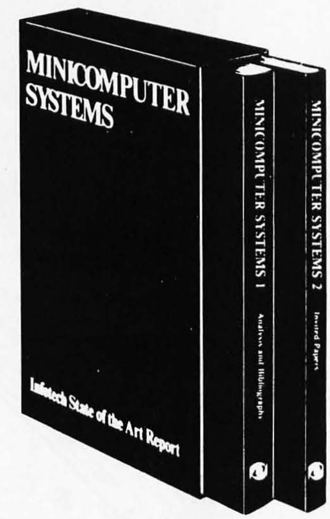
I feel that the efforts of an implementor of an 8080 APL interpreter should be directed toward writing a full set of APL that is reasonably memory efficient. This is not as hard as it seems: There are a few features that can be built into the interpreter which can reduce the memory requirements of an APL function down to a reasonable level.

The first problem to solve is the number of bits of resolution needed for accurate data representation. If a person is doing floating point calculations, he or she should have a full 16 digits of precision available (ie: eight bytes of information per numeric value). On the other hand, it is quite extravagant to use 64 bits for storing Boolean or character data. Also, operations on integer data (such as loop variables, counters, etc) should be carried out using a small, reasonably fast integer format, rather than a slow floating point format.

For example, in Microsoft's upcoming APL interpreter, we have chosen to store data in four different formats: 1 byte unsigned integer, 2 and 4 byte signed integer, and 8 byte floating point. These types are transparent to the user, and any necessary

AUERBACH Publishers, exclusive distributor for INFOTECH INTERNATIONAL
introduces the

COMPUTER STATE OF THE ART REPORTS



1977 Series
a unique blend of theory and practice in computing

Computer professional . . .

Keeping up-to-date and in touch with your ever-changing technology is as close as the *COMPUTER STATE OF THE ART REPORTS* from INFOTECH INTERNATIONAL. These unique publications are:

- Complete in-depth surveys of all *current, critical* and *dynamic areas* of computing
- Capable of influencing the *nature* and *direction* of computing throughout the next decade
- *Seminal works* in their *own fields*

Drawing on the finest and most erudite sources in the world, each INFOTECH Report is a comprehensive, analytical study of the most current and critical technical information in the data processing field today. The new 1977 Series covers . . .

33. Minicomputer Systems
34. Software Engineering Techniques
35. Microprocessors
36. On-Line Data Bases
37. Software Reliability
38. Future Systems
39. Distributed Processing
40. Performance Modeling and Prediction

Emanating from international conferences on advanced computer technology, each Report includes . . .

A unique method of *Analysis* — a distillation and synthesis drawn from formal presentations and question and answer sessions of in-depth seminars.

Invited Papers — especially commissioned from worldwide leading experts in their fields — which have been selected to give comprehensive coverage to each Report's area of expertise.

Bibliography and *Indexes.*

Published in one calendar year, these eight information-filled Reports are just $1295 . . . individual Reports are $215 each. Also available at these prices are the 1974/5 Series and the 1976 Series.

AUERBACH Publishers Inc., a leading source of data processing and data communications material and information, is proud to make these Reports available to you, and urge you to join most major computing organizations worldwide who are taking advantage of the valuable information service.

Save money and time and stay right up-to-date with INFOTECH INTERNATIONAL'S *COMPUTER STATE OF THE ART REPORTS.* To order your Reports or receive your free 28 page catalogue . . . fill out this coupon and mail to . . .

AUERBACH ®

AUERBACH Publishers Inc.
6560 North Park Drive
Pennsauken, N.J. 08109

INFOTECH COMPUTER STATE OF THE ART REPORTS ORDER FORM

Name/Title ____
Company ____
Address ____
City ____ State ____ Zip ____
Phone (Area Code) ____
Authorizing Signature ____
☐ Send me the 1977 Series at $1295.
☐ Send me the following Individual Reports at $215: # ____ , # ____
____ , # ____ , # ____ , # ____ , # ____ , # ____
☐ Please send my free 28 page catalogue about INFOTECH State of the Art Reports.

Make Checks Payable to:
AUERBACH Publishers Inc.

Check Enclosed: ____
P.O. Number: ____
Do you currently subscribe to any AUERBACH services?
☐ Yes ☐ No

AUERBACH

AUERBACH Publishers Inc., 6560 North Park Drive, Pennsauken, NJ, 08109.

B/777

Circle 136 on inquiry card.

conversions between types are performed automatically. If one adds two arrays, for example, there is no need to worry about the best format for the arrays or for the result. This multiformat scheme will also tend to increase the speed of the interpreter, since operations between integers are much faster than operations between floating numbers.

For this multiformat scheme to be effective, the interpreter must be as stingy as possible in converting to a larger format when deciding on the size of the result of a calculation. If two arrays are being multiplied, for example, the interpreter will assume that the size of the resulting array will be the same as that of the larger operand. Only when this assumption fails (ie: when the product of two elements of the arrays overflows the result type) does the interpreter increase the size of the result.

As careful as the interpreter is, occasionally a variable is stored in an overly large format. This can happen if the square root is taken of an array whose elements are all perfect squares, for example. To solve this problem the interpreter scans all data during terminal input waits and converts each one to the smallest format that can hold it.

Another storage saver that can be built into an interpreter is a careful examination of each statement to eliminate unneeded temporaries. For example, consider this APL statement:

$$Z \leftarrow X \times Y$$

The simplest way to operate on this statement would be to make copies of both X and Y, multiply them, and then store the result in a new location pointed to by Z. While this is an acceptable way of handling scalars, it is wasteful to the extreme if X and Y are large arrays. This statement should use only enough memory to hold the result and the two operands. No temporaries are necessary. Accomplishing this is not difficult when the interpreter knows how large the result will be; it is a more difficult problem when the type of the result is uncertain. If both X and Y are stored in a 1 byte format, for example, Z might require either one or two bytes per value depending on the contents of X and Y. Since the final size of the result cannot be determined until the last elements of the arrays are multiplied together, it might seem necessary to go back and start again using a 2 byte multiply routine!. A more sane solution, though, is to go back and convert all previous results from 1 byte to 2 byte format, and then continue from

the point of overflow using 2 byte multiply. Since upward type conversion is very fast, the overhead on overflow will be small.

Two other ways of saving storage are worth mentioning: data sharing between variables, and the overwriting of temporary results. Data sharing, the more difficult of two to implement, is a method of saving storage by keeping as few copies as possible of any one set of data. For example, if A is an array, then the statement B ← A should not copy any data: B and A should point to the same array in memory. The sharing will continue until either A or B is modified, at which point the interpreter is forced to make an extra copy of the data. Although this example is rather contrived, the technique can be of great value when calling functions. Only rarely does a function modify its arguments (SORT being a notable exception), so there is little point in making unnecessary extra copies of them. Data sharing has the additional advantage of requiring very little overhead, just one byte per data structure counting the number of variables pointing toward it.

The technique of overwriting previous results can also be a big scratch memory saver. Consider this APL statement:

```
B ← ÷ 1 + A
```

Let us assume that A is an array stored in floating point format. Working from right to left, the interpreter adds 1 to A, then takes the reciprocal of that result and assigns it to B. At first glance it might seem that this statement requires two temporaries: a place to store 1 + A, and a place to store the reciprocal of 1 + A. The interpreter can avoid creating the second of these temporaries by writing the result of the reciprocal operator over the 1 + A temporary. The only remaining problem is finding a quick way to distinguish between a "true" array, pointed to by a named variable, and a temporary array. The solution is to look at the reference count of the array: If it is zero, then no variable is pointing toward it and it can be safely destroyed.

No matter how careful a programmer is, or how many tricks he or she builds into an interpreter, there is no way of avoiding the fact that implementing a powerful language like APL on a small computer will take a lot of memory, certainly more than a BASIC interpreter. As I consider all of APL's useful features (and the plummeting of memory prices), I feel sure that APL will become one of the most popular languages among small system enthusiasts. APL programs are short and easy to understand once you are familiar with the Greek character set.■

An 8 Digit Hexadecimal Readout

R R Burns
17502 Teachers Av
Irvine CA 92714

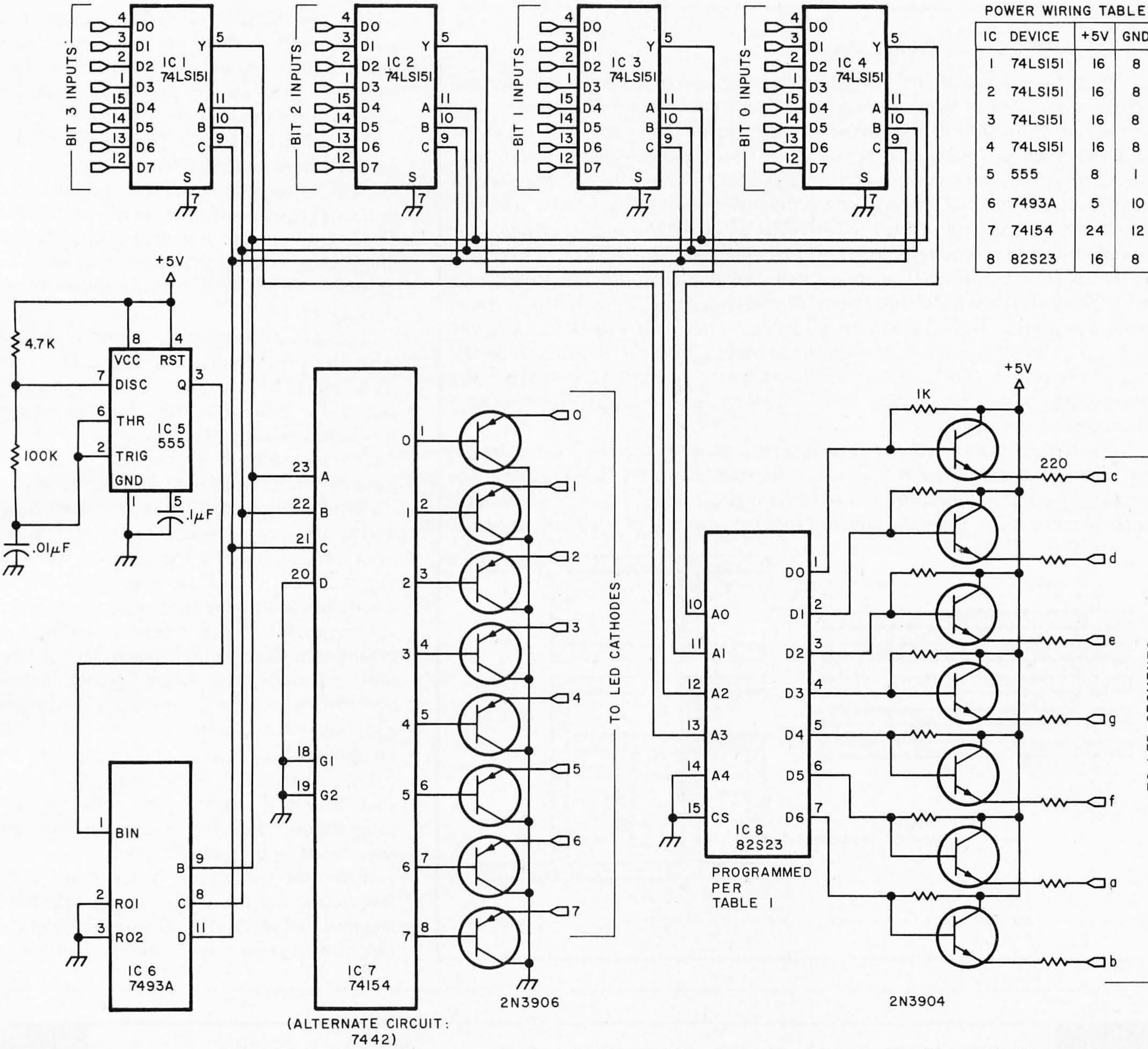
While working or playing with a microprocessor the logic state of many lines must be studied simultaneously in order to understand and debug the system. Typically, 16 address lines, eight data lines and possibly some IO lines are of interest. Each line can be individually checked with a meter or oscilloscope, but it is much more convenient to measure the states of the lines in parallel. Most microprocessor front panels have many LED indicators to simultaneously read out the address and data lines.

Table 1: The truth table of the read only memory that is used for the display. With this table it is possible to have a separate, distinguishable character for each of the hexadecimal digits.

Hexadecimal Decoder Address	b a f g e d c 7 6 5 4 3 2 1 0	segment bit
00	0 1 1 1 0 1 1 1	0
01	0 1 0 0 0 0 0 1	1
02	0 1 1 0 1 1 1 0	2
03	0 1 1 0 1 0 1 1	3
04	0 1 0 1 1 0 0 1	4
05	0 0 1 1 1 0 1 1	5
06	0 0 1 1 1 1 1 1	6
07	0 1 1 0 0 0 0 1	7
08	0 1 1 1 1 1 1 1	8
09	0 1 1 1 1 0 0 1	9
0A	0 1 1 1 1 1 0 1	A
0B	0 0 0 1 1 1 1 1	b
0C	0 0 1 1 0 1 1 0	C
0D	0 1 0 0 1 1 1 1	d
0E	0 0 1 1 1 1 1 0	E
0F	0 0 1 1 1 1 0 0	F
10 to 1F	All zeros	blank

Since microprocessors are byte oriented machines, hexadecimal digits are often used when writing code. After countless mistakes decoding data and address lines, I knew I needed an automatic method to translate the many ones and zeros to hexadecimal. The 16 hexadecimal digits 0 to 9 and A to F can be displayed in a directly readable format using a 7 segment readout. A check of the data sheets for the common TTL binary coded decimal to 7 segment decoders, such as the 7447, showed none display the letters for inputs greater than nine. Although the display may be unique for each input, a directly readable digit is required or the system is not much of an improvement over individual lamps.

A simple solution is to build a custom decoder by burning part of an 8 bit by 32 word bipolar programmable read only memory. The hexadecimal digit to be displayed provides four of the five address bits that determine the decoder's output. Seven of the eight output bits are used to turn on or off the appropriate segments of an LED readout. Table 1 shows the truth table for the decoder. It can be programmed using methods such as those described on page 66 of the November 1975 BYTE or by the distributor at the time of purchase. The fifth input, highest order bit, is grounded since only half of the read only memory is needed to hold the hexadecimal 7 segment code. The eighth output bit is also unused. It could be used to light the decimal point during display of input values greater than nine to emphasize the hexadecimal nature of



POWER WIRING TABLE

IC	DEVICE	+5V	GND
1	74LS151	16	8
2	74LS151	16	8
3	74LS151	16	8
4	74LS151	16	8
5	555	8	1
6	7493A	5	10
7	74154	24	12
8	82S23	16	8

Figure 1: This is the schematic diagram for the 8 digit hexadecimal readout. The output for the display is multiplexed to save on the cost of the relatively expensive read only memory. All resistors are .25 W and are measured in ohms.

the readout. Since only minimal confusion is present (b could be read as 6), this was not felt to be necessary.

The circuit diagram is shown in figure 1. Since programmable read only memories are relatively expensive, the display is multiplexed to reduce cost. This also results in a power savings. A 555 timer oscillates at about 1000 Hz and drives a 7493 which is used as a 3 bit counter. Its output address determines which input digit is to be displayed by selecting the correct output from each of the 74LS151 multiplexers and also sets a 74154 decoder to drive the corresponding readout digit. Each 74LS151 multiplexes one bit of the eight different input digits. Low power or low power Schottky should be used to minimize line loading of the device under test. The four bits to be decoded, the outputs from the multiplexers, address the 82S23 decoder as discussed above. Common cathode 7 segment LED readouts are interfaced to the decoder outputs and to the 74154 outputs using PNP and NPN transistor drivers.

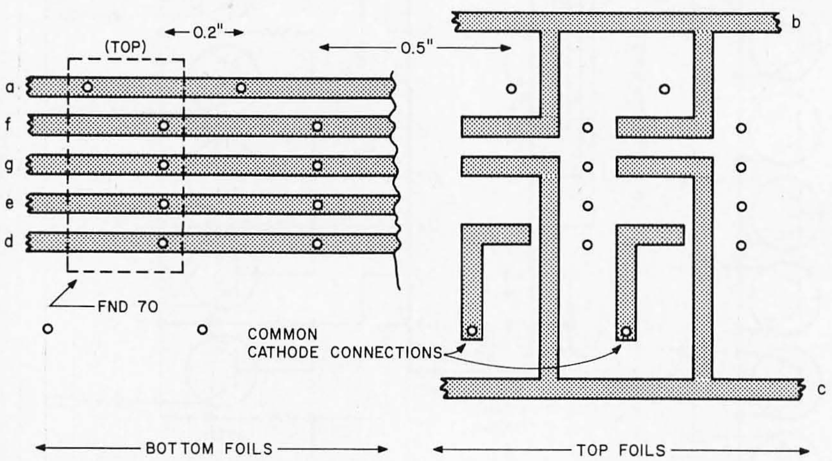
The component layout is not critical and the construction technique will depend on the use intended. The unit could be built as a dedicated front panel; however, since I feel that a microprocessor should be brought up with a monitor system that enables the system video display or teleprinter to serve as the front panel, I built the readout to be a stand alone tester.

The timer circuit, transistor and resistor networks, and the output leads from the

How to Make a Display Printed Circuit . . .

The pattern below shows the typical foils for two positions of hexadecimal display fabricated at home with printed circuit techniques. This double sided board can be made by first laying out all the holes for all the displays. (A template made by marking a section of "P" pattern Vectorboard can be used to guarantee the same pattern at each position.) Then using the template or hole locations marked without the template, the set of six holes for each position is drilled with a small (eg: #60) drill. After thoroughly cleaning the board, the pattern is laid out using printed circuit resist tape, a resist pen such as the Sanford's "Sharpie" marking pen available at stationery stores, or a resist paint. Both sides of the board receive this hand crafted pattern for as many positions as you care to wire up. When mounting the LED readouts, the b, c and cathode pins are bent outward to make contact with the pads on the top of the board. Due to the difficulty of removing the LED readouts after installation, testing the devices prior to soldering is a highly recommended procedure.

In this figure, the patterns for only two readout sections are illustrated; the hole and circuit pattern is repeated for each additional one required. The segments are lettered near the printed circuit paths which connect them. Both patterns are viewed from the top foil looking into the printed circuit board.



74154 and 82S23, as well as voltage and ground, were etched on a printed circuit board. The rest of the circuit was wire-wrapped. Two 16 pin dual in line package sockets were used for the input connectors.

Eight Fairchild FND-70 7 segment LED displays were mounted on a separate board in four groups of two to represent four bytes. The segments are wired in parallel. These readouts do not plug into a dual in line package socket and must be either point to point wired, or a printed circuit board must be made.

The readout is simply used by connecting it to the lines of interest. The data on the lines must be stepped slowly enough to enable the data to be read. For cases where this is not possible the data could be latched on command by a strobe that selects the time of interest and then read out at leisure.

Besides serving as a temporary front panel during the period when a new microprocessor system is being brought up, the readout is useful for later use to monitor IO lines during program debugging sessions. It is also a very useful tool for general testing and breadboard development when several lines must be monitored. Example uses include deciphering read only memories made long ago, but left unlabeled, checking out surplus integrated circuits such as comparators, decoders, adders, etc. It can be attached to the switches of a manual read only memory programmer to give insurance that the switches have been properly set.

From my experience, it is well worth the few hours and few dollars to get together a hexadecimal readout. You will soon find it is one of the handiest tools on your bench.■

You should subscribe to creative computing today! Here's why —

1. Creative Computing will help you decide which computer is best for you.

Creative's no-nonsense equipment profiles arm you with the facts before you talk to the vendor or dealer. Whether you're interested in a microcomputer kit, a mini, terminal, or programmable calculator, you'll find the evaluative information you need in *Creative*. Indeed, one wise hardware decision could save you the cost of a lifetime subscription!

2. Creative Computing discusses computer applications in non-technical, understandable language.

Business people who want to know what's going on in the EDP department, students who want to learn about microprocessors, hobbyists looking to make good use of home computers, or anyone concerned about the effect of the computer on society will find these and many, many more mind-expanding topics covered on the pages of *Creative*.

3. Creative Computing covers computer education in depth.

After all, that's where we got our start and so we continue to present four or five major learning activities every issue. If you're a teacher, *Creative* will save you hours of preparation time. If you're a student, you'll be way ahead of your class with *Creative*. And if you've already graduated, you can bone up on what you missed.

4. Creative Computing carries outstanding fiction every issue.

One of the best ways of exploring future scenarios of computer usage is through fiction, so *Creative* seeks out material from the best authors — Isaac Asimov, Frederik Pohl, Arthur C. Clarke to name just a few, as well as many others who are destined to be the best of the next generation.

5. Creative Computing's book reviews are honest and timely.

We're not owned by a big book publisher to whom we owe loyalty, nor do we depend upon advertising for our revenue. Hence, not only do our reviews pull no punches, but we also rank order similar books (like all 34 books on the BASIC language which we reviewed last year). *Creative* reviews virtually every computer book of interest to educators, hobbyists, and general laypeople, even including vendor manuals and government pamphlets.

6. An extensive resource section will save you time and money.

Every issue of *Creative* carries 40 or more short resource reviews evaluating periodicals, booklets, hardware, terminals, couplers, peripherals, software packages, organizations, dealers, and much more. Every entry has a brief description, evaluation, and the name, address, and phone number of the vendor. You'll save valuable time seeking out this information, much of which you'd possibly never come across.

7. Creative Computing will provide hours of mind-expanding entertainment, even if you don't have a computer.

Creative Computing carries 10 or 12 pages of games and puzzles every issue. Most of the puzzles don't need a computer or calculator to solve; some do. Naturally, the 4 or 5 new computer games (in Basic, Fortran, and APL) in every issue require access to a computer.

8. Creative Computing gives you things to actually do with a computer.

Home computer kit, mini, timesharing terminal — whatever your access to computer power, *Creative* provides thoroughly documented programs with complete listings and sample runs that you can use with minimum effort. Games, simulations, CAI, computer art — whether education or recreation is your bag, you'll find programs which you can use in *Creative*.

9. A no-compromise policy of editorial excellence means every issue is of value to you.

We firmly intend to be around a long time and we believe the way to do that is to publish only material of the very highest quality. We believe our readers are among the most astute, intelligent, and active people in the world. You can't afford to waste time reading imprecise, opinionated, or wordy articles and you won't find any in *Creative*.

10. The price is right — only $21 for 3 years.

That same $21 will buy you a pair of Star Trek walkie talkies, six direct dialed 10 minute calls between New York and Boston, 3 tankfulls of gas, or 10 cocktails at a Hilton hotel. Wouldn't you rather have 18 issues of Creative Computing each containing over 85 pages of solid editorial material (including advertising, over 100 pages per issue). Count the editorial pages in *any other* hobbyist or special interest magazine and compare it to *Creative*. *Any other*. 1 year subscription $8. Lifetime $300.

NO RISK GUARANTEE

You may cancel your subscription at any time for any reason and we will refund the balance without question.

David H. Ahl, Publisher

FOR FASTER RESPONSE

800-631-8112

(In NJ, call 201-540-0445)

SUBSCRIPTION ORDER FORM

Type	Term	USA	Foreign
Individual	1-Year	□ $ 8	□ $ 11
	3-Year	□ 21	□ 30
	Lifetime	□ 300	□ 400
Institutional	1-Year	□ 15	□ 15
	3-Year	□ 40	□ 40

□ New □ Renewal

□ Cash, check, or M.O. enclosed

□ BankAmericard Card No. ____________

□ Master Charge Expiration date ____________

□ Please bill me ($1.00 billing fee will be added)

Name ____________________

Address ____________________

City ____________ State ______ Zip ______

Send to: Creative Computing, Attn: Darcy
P.O. Box 789-M, Morristown, NJ 07960

DO IT WITH 16 BITS

DO IT BETTER

With the language that is best suited for your application: Business programming in **COBOL**, Scientific and Engineering in **FORTRAN IV**, Educational in **BASIC**, and Systems Implementation in our **MACRO ASSEMBLER.**

With hardware that will grow with your application, and protect your software investment.

DO IT FASTER

With extensive support and utility programs to speed development of your applications software. These include a Diskette Operating System, Relocatable Linking Loader, Load Module Library Editor, Symbolic Debugger, and Text Editors.

With a 16 bit processor that includes hardware multiply and divide, real-time clock, and peripherals like single or dual diskette subsystems with integral DMA controller.

DO IT WITH A MicroNOVA®

BPI

If you are serious about computing, call or write today. BPI Inc., 2205 East Broadway, Suite 6, Tucson, Arizona 85719 (602) 326-6975

® NOVA is a registered trademark of Data General Corporation, Southboro, Massachusetts

Circle 141 on inquiry card.

Look What You Can Do...

...with an Edge as a Cue

Ralph Tenny
POB 545
Richardson TX 75080

There are hundreds of digital integrated circuit functions, some highly specialized and others perfectly general. So long as simple digital functions are involved, logic cookbooks, truth tables and functional descriptions explain what various integrated circuits do and show how to apply "standard" solutions to problems. But it is often possible to combine integrated circuits and other components to provide clever solutions to various problems, taking advantage of the characteristics of the integrated circuits. In this article, we concentrate upon several different ways to generate pulses or time delays on cue, using analog, asynchronous and synchronous (clocked) digital methods.

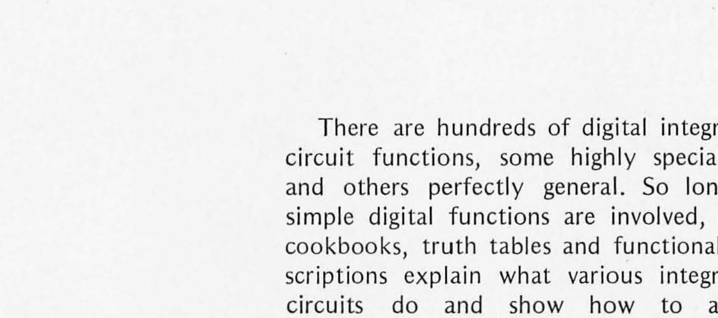


Figure 1: Comparison of the Schmitt Trigger and Linear Amplifier. The simple linear amplifier generates an output voltage which is a fixed multiple of the input voltage. The graph at the left shows what happens to a simple input waveform as amplified with a gain greater than unity. The Schmitt trigger, in contrast, has two threshold levels, V1 on the rising side and V2 on the falling side of the input waveform. The output changes from a high logic level to a low logic level when the V1 threshold is crossed, and returns to the high logic level when the V2 threshold is crossed in the opposite direction. In this example a 7414 circuit is used, which is an inverting Schmitt trigger.

Exploring the Schmitt Trigger

Schmitt trigger integrated circuits are available in both the TTL and CMOS logic families, and share a common characteristic: A property called hysteresis has been incorporated in the circuitry between input and output, which leads to a useful time delay effect with signals that are slowly varying compared to logic speeds. This action can be illustrated by comparing a linear amplifier's response with the response of a Schmitt trigger. Figure 1a shows how the linear amplifier responds to an input waveform: It simply amplifies the signal but otherwise makes a faithful copy. In figure 1b, the response of the Schmitt trigger to the same type of input waveform is illustrated. Note how the Schmitt trigger output makes no change until the input crosses a threshold V1, then makes no further change until the input voltage has dropped to a lower threshold, V2. The hysteresis property of the Schmitt trigger is this difference between the rising (V1) and falling (V2) change thresholds. The use of this hysteresis to reduce noise on a signal is illustrated graphically in figure 2. The analog input signal in figure 2 would cause problems if applied to a TTL gate input. By simply taking advantage of the hysteresis levels V1 and V2 using a Schmitt trigger, the input signal is transformed into a much more usable signal with clean (ie: the typical 10 ns) transitions.

Because the Schmitt trigger's hysteresis rejects noise so readily and has a rapid response once the threshold has been crossed, it is often used to accept slow waveforms for input to normal logic cir-

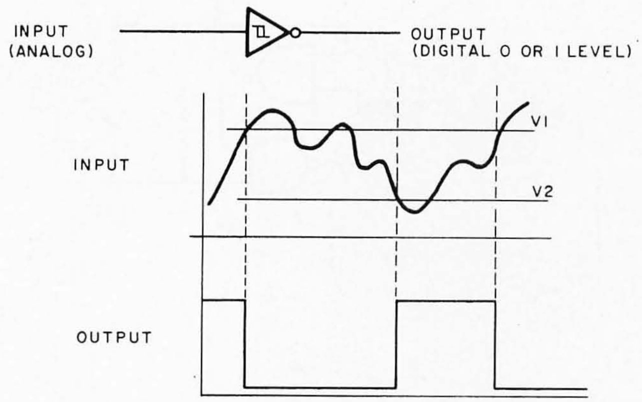


Figure 2: One of the principal uses of the Schmitt trigger circuit is to convert slowly varying analog signals into the nearly instantaneous transitions of digital logic – and at the same time to filter out minor variations. This example illustrates how an arbitrary analog input is converted into a digital signal by a Schmitt trigger with thresholds V1 and V2. Changes only occur at V1 when the input is rising, and V2 when the input is falling.

cuitry. With proper external circuitry, the Schmitt trigger will also accept capacitively coupled waveforms. Due to this property, the Schmitt trigger is a very versatile device.

An example of the Schmitt trigger's application is the generation of a trigger pulse on an "edge" of another waveform. Figure 3 shows two different edge detectors made with the 7414 circuit as the key Schmitt trigger element. Figure 3a shows the positive edge detector, and Figure 3b shows the negative edge detector. The Schmitt trigger's properties of hysteresis allow the reliable use of long time delays in the RC network driving the device; use of an ordinary TTL gate would not be reliable due to the optimization of TTL devices to standard TTL transition speeds. The timing diagram in figure 3 shows the relationship of the two edge detector outputs to a common driving waveform. How about pulse delays? Cascading several such edge detectors gives a series of delayed pulses as shown in figure 4. The timing is determined by the RC components used, and is thus called "asynchronous" since it has no dependence upon a regular clock period and can vary with the temperature and other physical parameters which affect resistors and capacitors.

The Schmitt trigger also interfaces with the real world outside digitland. Figure 5 shows two ways to create what is called a "power on reset." Such a reset function is used to clear digital logic such as counters, sequencers, controllers, and microprocessors which must start from a predetermined state following the application of power to the system. A typical rule to thumb is to choose the RC time constant to give a reset delay at least three times as long as the power line takes to come up to full voltage.

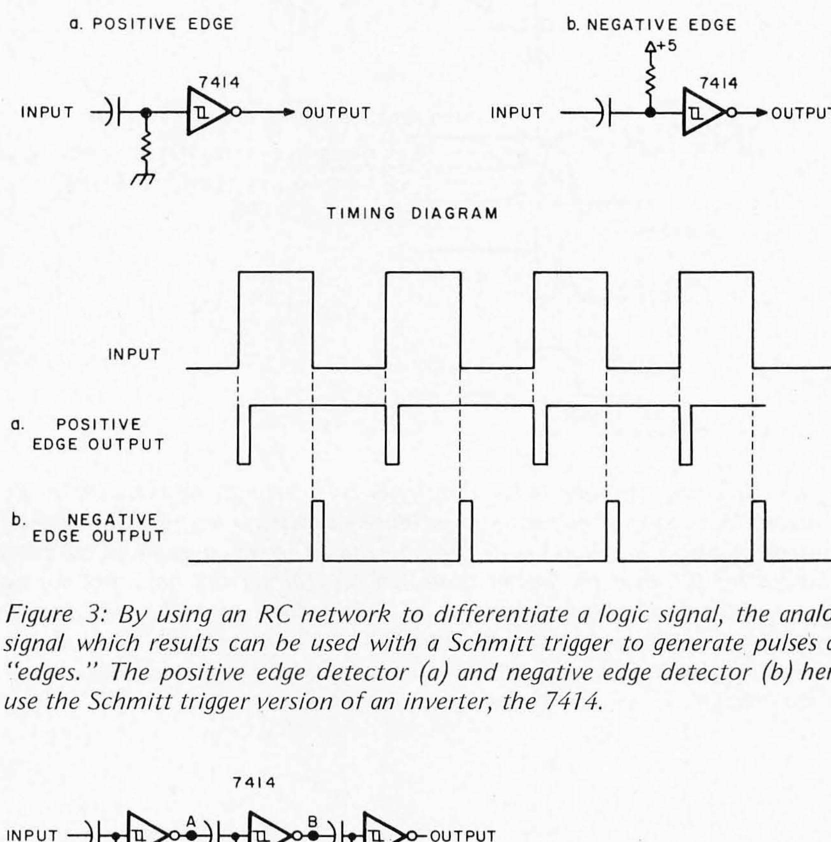


Figure 3: By using an RC network to differentiate a logic signal, the analog signal which results can be used with a Schmitt trigger to generate pulses at "edges." The positive edge detector (a) and negative edge detector (b) here use the Schmitt trigger version of an inverter, the 7414.

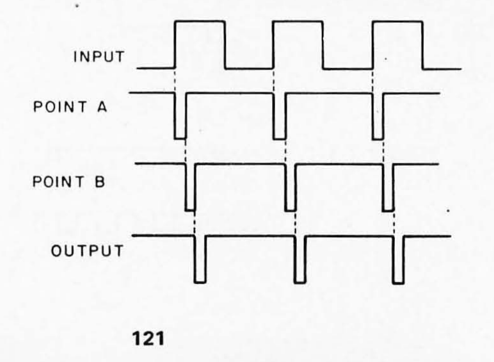
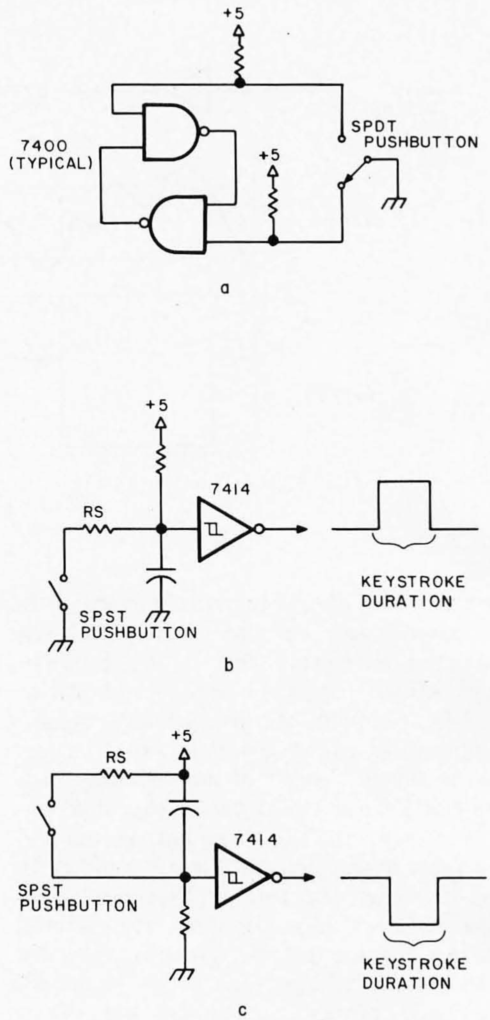


Figure 4: A cascading series of positive edge detectors can easily be made using several Schmitt trigger stages. Each stage uses the logic signal generated by the previous stage, resulting in a series of pulses with progressively longer delays relative to the input signal's rising edge.



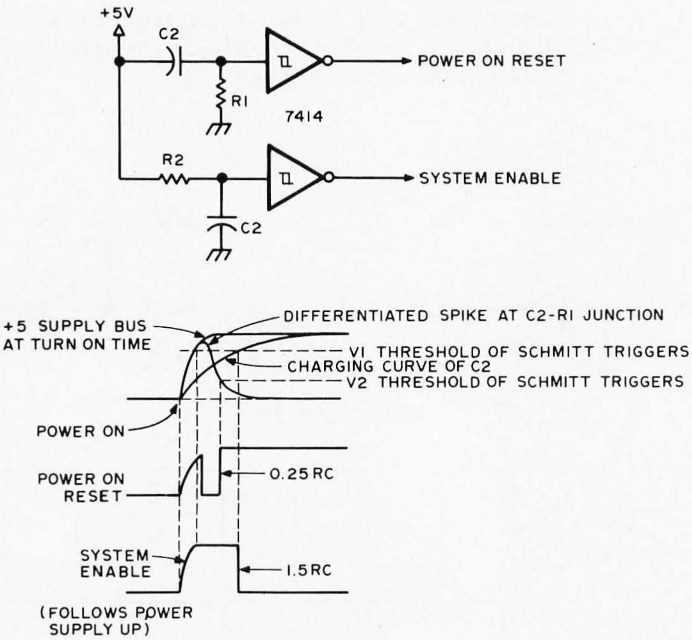


Figure 5: Using the time delay properties of a Schmitt trigger with an RC network leads to this technique to produce a power on reset signal. A master system enable (a low state) is delayed by the RC time constant of C2 being charged by R2 after the power is turned on. During the time that normal system action is disabled, the power on reset pulse is created by the positive edge trigger of the upper Schmitt circuit. The RC constants are chosen so that the system enable falling edge follows the rising edge of the power on reset by a good margin.

Figure 6: The time delay properties of a Schmitt trigger and RC networks can be used to debounce switches. The common approach is to use a flip flop such as the circuit at (a). However, by using a Schmitt circuit, it is possible to get away with an SPST switch instead of the SPDT switch required for the flip flop technique. Two versions of SPST switch debouncing are shown at (b) and (c).

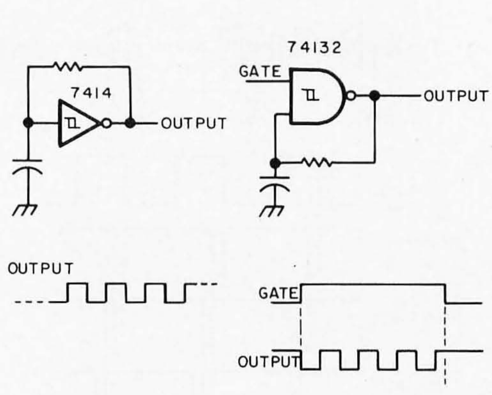


Figure 7: An inverting Schmitt trigger can be used as an oscillator by feeding its output back to its input with an RC network as shown at (a). If a Schmitt trigger NAND gate is used instead of the inverter form, the extra input can be used to gate the oscillator on or off.

One of the most troublesome real world logic interface problems is switch debouncing. Any switch contact bounces, a little or a lot depending upon many factors. A typical way of debouncing a switch is to use two NAND (or NOR) gates in a set-reset flip flop circuit as shown in figure 6a. However, this circuit requires a single pole, double throw switch. The circuits of figures 6b and 6c show how the Schmitt trigger gate can be used to generate a debounced TTL signal from an SPST switch by using RC components to generate a time delay which is longer than the typical bounce interval for the contacts.

One other place the Schmitt trigger shines is in the digital oscillator as shown in figure 7a. Besides the 7414 hex inverter package, Schmitt triggers come in several other packages including a quad NAND package called the 74132. As an oscillator, the 74132's gating action can be used to gate the oscillator on or off using the second input as shown in figure 7b. (Note that any of the previous illustrations can be implemented with the 74132 or other gates in the 7400 series of parts with Schmitt trigger inputs.)

For those who may not have the appropriate logic handbooks, let's note that the two Schmitt trigger integrated circuits mentioned above are functionally identical to their normal TTL counterparts, with the exception of the Schmitt input property. In figure 8, these pinouts are reproduced for reference purposes. If you need a spare inverter or NAND gate, the Schmitt trigger property of any extra active sections can be ignored without side effects in most cases.

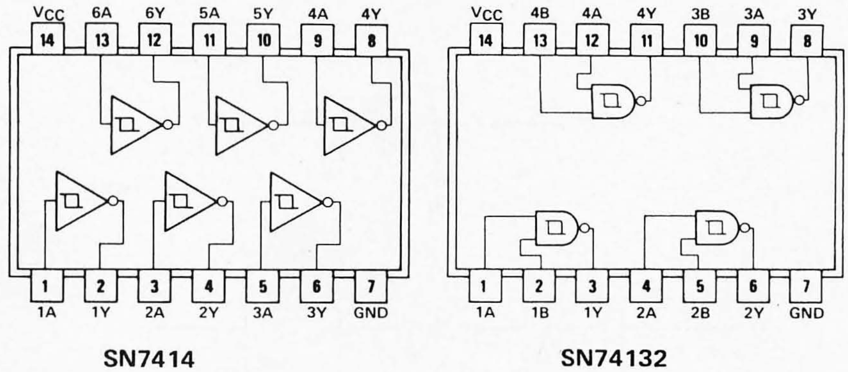


Figure 8: Some Pinouts for Review. This illustration shows the pinouts of the 7414 and 74132 Schmitt trigger packages. Note that these pinouts correspond exactly to the pinouts of the 7404 and 7400 respectively. The only difference between the standard inverter and NAND gates and the Schmitt trigger forms is the fact that the inputs have the Schmitt trigger property. These figures reproduced from pages 5-9 and 5-37 of the TTL Data Book for Design Engineers *(available through BITS).*

The Schmitt trigger aspect of these gates is thus a useful bonus. Every digital kluge board will most likely require at least one Schmitt trigger integrated circuit on it for those unexpected noise and timing problems which crop up sooner or later. One other item should be noted about the TTL Schmitt triggers: Pull down resistors and feedback resistors (see the examples in the figures) should not be larger than 390 ohms. Pull up resistors can range up to about 2.2 k ohms.

A Schmitt trigger can be made using positive feedback around an AND gate.

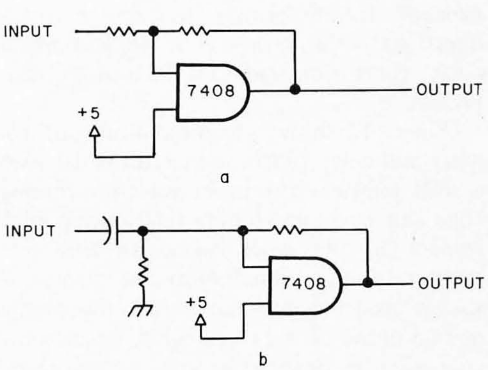


Figure 9: If the Schmitt trigger property is needed in an existing circuit, an AND gate can often be modified as a temporary expedient. The circuit at (a) shows a 7408 AND section used as the equivalent of the Schmitt trigger in figure 1b. The circuit at (b) shows the equivalent of the positive edge connector in figure 3a.

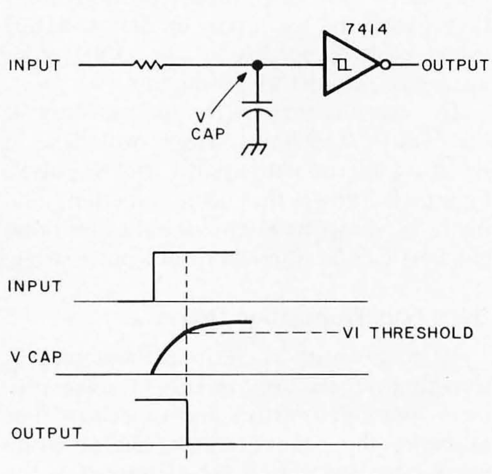


Figure 10: If the input network of the Schmitt trigger is hooked up as an integrator, charging the capacitor, the pulse edge out of the Schmitt circuit will be delayed. In this case, the negative edge of the 7414 output is a delayed and inverted copy of the positive edge input.

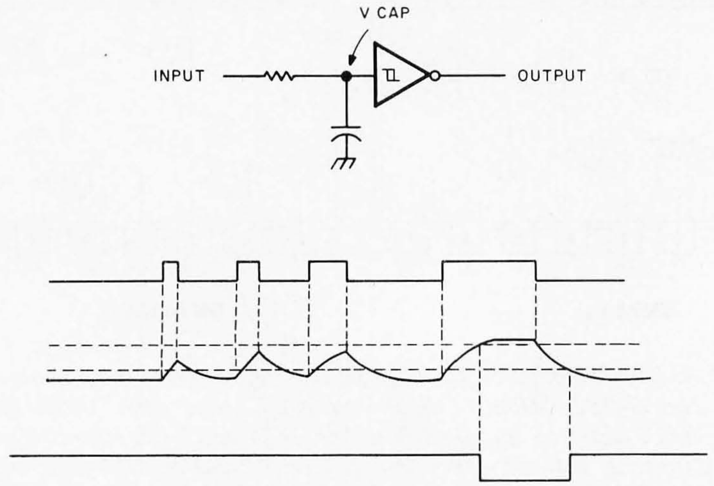


Figure 11: If a delay network such as that in figure 10 is presented with a string of short pulses, there will be a certain minimum length input which will be recognized as a transition on the output. If the pulse is not long enough to charge the capacitor to the threshold level through the resistor, then there will be no change at the output. Thus the Schmitt trigger with an RC network can be used to discriminate against short pulses.

Figure 9a shows the 7408 used as the basic Schmitt trigger in a circuit analogous to figure 1b. Figure 9b shows the 7408 substituting for the 7414 in a circuit like figure 3a. In both cases, except for the fact that the output is not inverted by the 7408, the effect is the same. Note that the unused input of the AND gate is tied to +5 V to enable the gate. Make the feedback resistor about 10 times the pull down or pull up resistor for best results. Use your own imagination to design all the previous functions (with the exception of the oscillator) using positive feedback and AND gates instead of Schmitt trigger circuits.

By reversing the resistor and capacitor at the inputs, a Schmitt trigger will delay a signal or discriminate against narrow pulses. Figure 10 shows the delay function, and figure 11 shows how narrow pulses (or noise glitches) can be filtered out of a pulse string.

Using Gate Propagation Delays

When gates are as cheap as they are, it is sometimes more appropriate to solve problems without resistors and capacitors. For example, the edge detector can be made using gate propagation delays instead of the RC time delays of a Schmitt trigger. Figure 12 shows such an edge detector. For those who may not be familiar with the principle, another common name for this circuit is a "delay oneshot." The delay oneshot works this way (referencing figure 12): The input is initially low, which forces the output high. With three inversions, point C will be initially high. When the input goes high, the output will go low (after one gate delay) since point C is still high. Three gate delays after the edge, point C will go low forcing the output high again after one gate delay. A similar process takes place for the NOR gate version of figure 12b. The timing diagrams of figure 12 show how the gate delays lead to the pulse effect.

The name delay oneshot comes from the fact that an integrated circuit's output does not change instantaneously with its input; rather there is a finite propagation delay shown as ΔT in figure 12. Therefore point C goes low three propagation delays after the input went high, and the output will then be low (for figure 12a) for a short time of about 25 to 35 ns with standard TTL parts. The exact time depends upon the particular integrated circuit, its current temperature, and other physical parameters. In many applications, the exact pulse width is not important; for it is wide enough to trigger clock and edge sensitive inputs of other TTL devices such as flip flops and counters. While figure 12a shows a positive edge detector, using a NOR instead of a NAND as in figure 12b yields the negative edge detector. The circuit of figure 12c completes the picture by using an exclusive OR gate to detect both edges of the input waveform. This exclusive OR circuit is also known as a frequency doubler since it produces two pulses in each period of the input waveform.

Digitally Controled Delays

Is a longer pulse needed? Do you need a pulse with a programmable length? Is a special synchronous timing relationship needed? If the answer to any of these questions is yes, then use of clocked digital delay generation methods is a useful approach.

Figure 13 shows a combination synchronizer and delay generation circuit which uses a shift register. The input waveform falling edge can have any timing relationship with respect to the clock signal to the shift register. In the illustration, its change is shown occuring approximately in the middle of one phase of the clock cycle. Whenever it changes, the next active edge of the clock (marked by an upward arrow) will read the input data into the first stage of the shift register. Then seven clock periods later, the output stage of the shift register will change, since the 74164 used in this example has a total of eight stages. By using the NAND gate shown along with the inverter, the time during which the Q_H and Q_A outputs of the shift register disagree is detected to form the

pulsed output waveform shown in the timing diagram. Note that, assuming a 1 MHz clock, the width of the pulse generated by this circuit will be exactly 7 μs. The relative timing of the input change and the first clocked transition of the shift register determine the exact starting time of the pulse. By adding other NAND gates tied to different shift register outputs, it is possible to produce several different but synchronized pulses for the same input edge. Also, since the change information is passed from stage to stage by the clock, the output of any stage will have a change edge which is synchronized to the clock. The example of figure 13 uses a 74164 shift register; the same principle can be used with shorter or longer shift registers. Note also that to obtain a complete 7 μs pulse, the input must remain low for at least 8 μs (worst case relative timing).

How about a oneshot with a programmable pulse width? Figure 13 shows how the combination of a presettable counter with a latch gives a programmable oneshot. It operates this way: assume that no negative going input pulses have reached the circuit lately. The normal operation of the counter is to continuously count through 16 states, giving a borrow output (called the "end pulse" in the diagrams) whenever state 0 is reached. Thus after power is turned on, within 16 clock cycles the RS latch will have been reset such that the output line of the circuit is low.

Now suppose an input pulse of arbitrary duration reaches the circuit. This pulse forces the digital pattern on the four input lines (noted C3 to C0 corresponding to their binary weights) to be loaded into the counter. During the load pulse, which need not be synchronous with the clock, the counter is inhibited from counting and the RS latch is set so that the output is high. As soon as the load pulse ends, the counter can again count down, from the loaded state of 3 in the example. Eventually the counter reaches zero having exhausted the programmed count. This causes the borrow pulse to be emitted, resetting the latch again, thus ending the generated pulse. The net length of the pulse generated is equal to the length of the input pulse plus the delay between the end of the input pulse and the end of state 3, plus 2.5 clock periods. If the input pulse is restricted to a short width

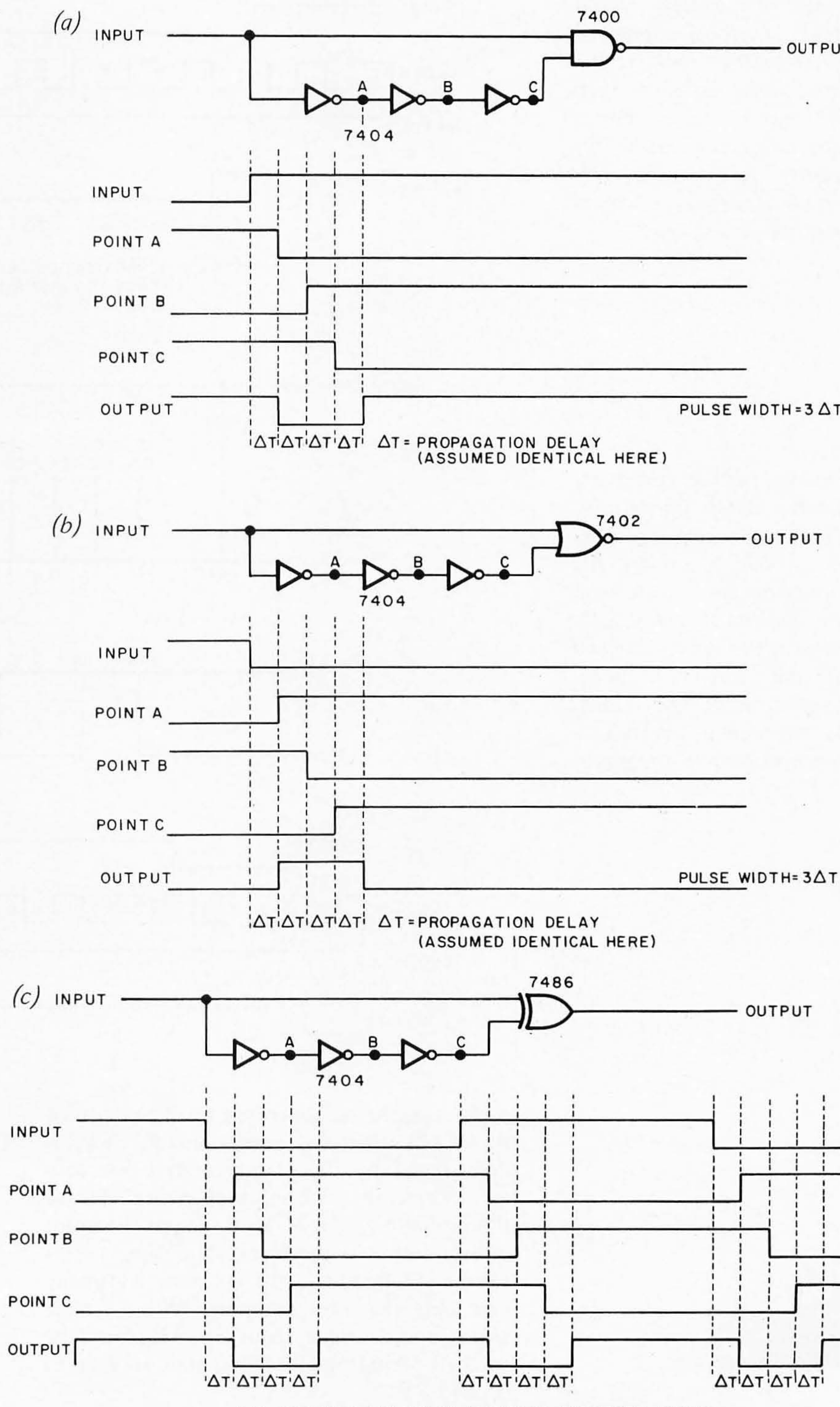


Figure 12: Making use of the finite (typically about 10 ns) gate delays of logic circuits, it is possible to make edge detectors without any analog components such as resistors or capacitors; (a) is a positive edge detector, (b) is a negative edge detector, and (c) is a frequency doubler, which detects both edges of a repetive logic waveform.

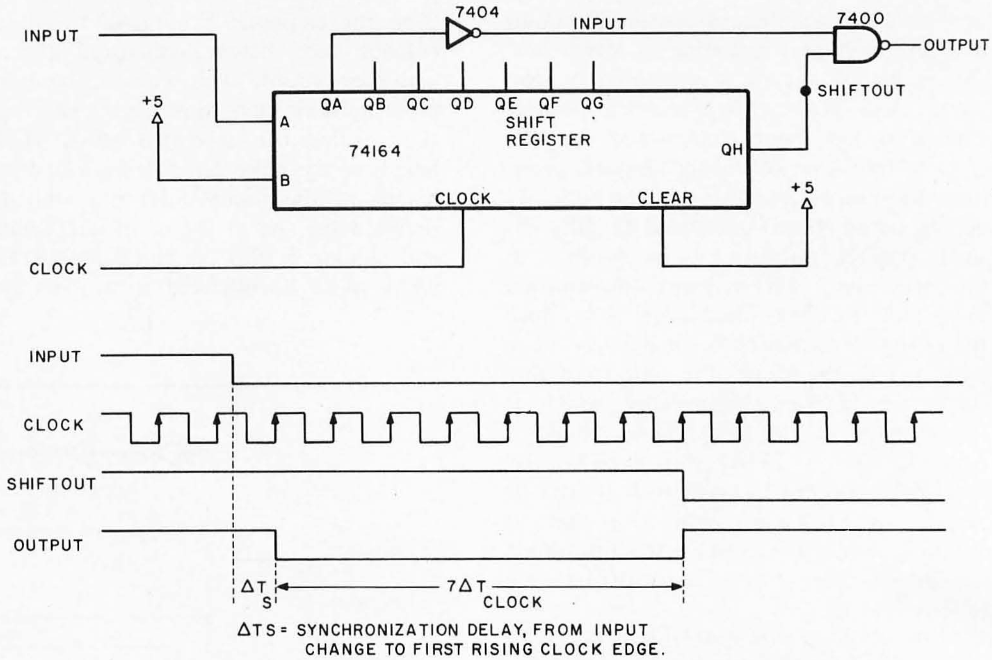


Figure 13: Use of digital circuits with clocked transitions enables delays, edge detection and other functions to be done with outputs synchronized to the clock. For example, this 8 bit shift register is used to obtain a pulse width of exactly seven clock periods, synchronized to the clock. If the input pulse is longer than the seven clock periods plus one clock period for relative timing slop, this circuit produces a digital negative edge trigger.

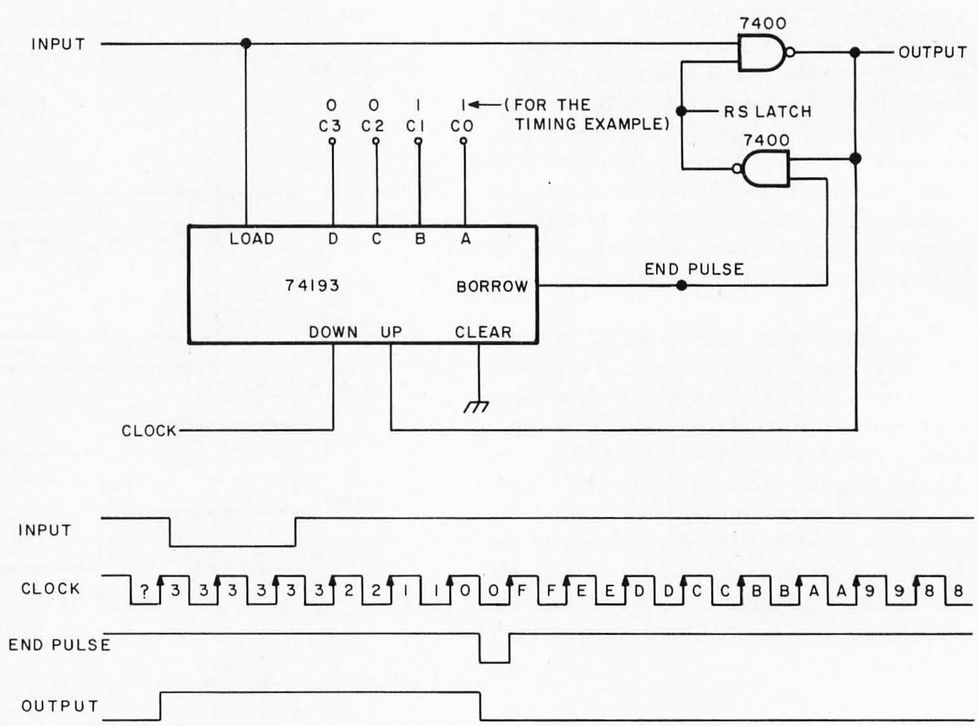


Figure 14: A programmable delay generator. This circuit generates a delay which is set by the inputs of the counter and the width of the trigger pulse at the input. The end of the pulse is synchronized with the clock, but the start of the pulse is synchronized to the input.

such as might be generated by the circuit of figure 12a, then the programmable delay is determined by the count entered but with an uncertainty of one clock period due to the possible variations in timing of the input pulse relative to the clock. This same circuit can be extended to arbitrary pulse widths by cascading the 4 bit counters. With a 1 MHz clock, for example, four 74193s could be used to programmably time periods of up to 65,535.5 μs.

Hopefully your imagination has been jogged a little by the discussion of these edge related phenomena. A number of circuits which treat edge related phenomena were made by using the characteristics of some integrated circuits and a little imagination. These particular techniques may not be immediately useful; but when you need one of the functions described, they will come in handy. In your spare time, remember to study the various integrated circuits, not just the cookbook applications, but any imaginative extensions you can dream up. ■



- ANY NUMBER OF FILES MAY BE OPEN (IN USE) AT ONE TIME
- THE NUMBER OF FILES AND SIZE OF FILES IS LIMITED ONLY BY THE SIZE OF THE DISC
- MERGING FILES REQUIRES NO EXTRA DISC SPACE
- NO WAITING FOR THE DISC TO RE-PACK
- LONGER DISC LIFE — MORE EVEN DISC WEAR

Our Basic Floppy Disc System (BFD-68) must, in all modesty be called superb. It comes completely assembled with a disc controller that is plug compatible with the SWTPC 6800. In fact all our products use the 6800 standard SS-50 (Smoke Signal 50) bus used by SWTPC. The cabinet and power supply are capable of handling up to 3 Shugart Mini-Floppy Drives. One drive is included in the price of the BFD-68 and others may be added easily at any time. Or you may save money by ordering the dual-drive BFD-68-2 or triple drive BFD-68-3 (pictured). Price: BFD-68 $795, BFD-68-2 $1169, BFD-68-3 $1539, SA-400 Drive $390.

The BFD-68 includes our Disc Operating System Software. Our software provides for a soft-sectored disc format consisting of 128 bytes per sector, 18 sectors per track and 35 tracks per disc. The software provides direct commands to name and rename files, transfer memory to disc and disc to memory and to automatically jump to the starting location of any program loaded from disc to memory. The direct command names are: RUN, GET, GETHEX, CLOSE, SAVE, DELETE, APPEND, RENAME, COPY, LIST, FIND, LINK and PRINT. In addition, the Disc File Management subroutines are available to create files under your program control.

A bootstrap PROM is included on the controller board to initiate the Disc Operating System which loads into a 4K memory board located at 7000 or optionally at D000. Thus, you can be up and running from a cold start in just a few seconds.

SUPER SOFTWARE

Free patches are provided for SWTPC BASIC version 2.0 and Co-Resident Editor/Assembler. These patches allow the SAVE and LOAD commands to work with the disc or the cassette at your option.

SUPER EDITOR: Smoke Signal Broadcasting now has its own editor. It is a content oriented editor with string search and block move capability. Changes may be made by referring either to line number or string content or a combination of references. Naturally, it is designed for file transfers to and from the BFD-68. Price: SE-1 $29 on diskette.

ALL OUR PRODUCTS EXCEPT THE PS-1 ARE COMPLETELY ASSEMBLED.

SUPER ASSEMBLER: Inputs source code from file on the BFD-68 disc system and outputs object code to disc file. Assembly listings include alphabetized and tabulated symbol table. Price: SA-1 $29 on diskette.

Complete source listing included for both editor and assembler. Order both for $53 and save $5.

SMARTBUG — A CURE FOR MIKBUGITIS: A super smart Motorola-Mikbug replacement that preserves almost all Mikbug entry locations so your present programs will run without modification. Uses ACIA instead of PIA and includes many additional features including a software single-step trace command. A SMARTBUG listing is included and object code is provided on a 2708 free with each P-38 series board purchased. Source listing available separately for $19.50.

NEED A FULL SIZE FLOPPY? Our P-38-FF is a plug-in interface card to the ICOM Frugal Floppy™. It includes all the features of the P-38-I plus one 2708 EPROM containing the ICOM bootstrap software. Just plug the P-38-FF into your SWTPC 6800 and your ICOM into the P-38-FF and you're ready to use the Frugal Floppy and ICOM's 6800 software package. Price $299.

Our P-38 is an 8K EPROM board containing room for 8 2708's. Or, you may use it to hold up to 7 2708's plus your Motorola Mikbug or Minibug II ROM. The P-38 addressing is switch selectable to any 8K location. Price $179.

The P-38-I contains all the features of the P-38 plus an interface to the Oliver Paper Tape Reader and our EPROM Programmer. Price $229.

The PS-1 Power Supply Kit provides plus and minus 16 volts required for the P-38 series boards. Also, it allows a wiring modification to be made to the 8 volt supply that will increase its output by one volt. Price $24.95.

Our M-16-A is a 16K single power supply STATIC RAM memory system. The M-16-A is fully buffered and requires only half the power of a similar size system using low-power 2102's. With the M-16-A, you can expand your system to 48K and still have room left for one of our EPROM boards. The M-16-A is switch selectable to any 4K starting address and hardware write protect is included. Quick delivery. Price $595.

BANKAMERICARD, VISA AND MASTER CHARGE WELCOME.

SMOKE SIGNAL BROADCASTING

P.O. Box 2017, Hollywood, CA 90028 • (213) 462-5652

Circle 142 on inquiry card.

BYTE's Bits

Getting Back to BASIC

If at first you don't succeed, try reentering your program! That was the order of the day at America's first large scale collegiate BASIC contest held at Taylor University, Upland IN, on April 23 1977. Taylor hosted teams from almost 50 midwestern colleges during a four hour struggle to solve five problems using Taylor's DEC 11/40 timesharing system. Taylor's team came in first with DePauw University, Greencastle IN, a close second.

Additional competitions are planned for next spring throughout the country, all to be hosted by Taylor. Interested teams should contact Professor R Waldo Roth, Chairman of the Computer Science Department, Taylor University, Upland IN 46989.■

Trenton Computer Festival

On the last weekend in April the Second Trenton Computer Festival was held at Trenton State College in Trenton NJ. The festival consisted of approximately 30 exhibitors, a large flea market, and a large number of technical talks which covered such varied subjects as robots, computer music, games and applications.

Several door prizes were given away; among these was a lifetime subscription to BYTE. The exuberant winner of this prize was Mel Richman of Philadelphia PA. Mel has homebrewed a 6502-based microprocessor on OSI boards. One of Mel's uses for his system is Morse Code transmission. The system is capable of sending Morse Code at four different speeds while displaying the characters being sent on a video screen.■

An Intensive Short Course on Personal Computers and Microcomputer-Based Business Systems

The Evening Division and Department of Mathematics and Computer Science of St John's University, Jamaica NY, has announced a series of intensive short courses on low cost personal computing. Each course is designed to provide an understanding and a well rounded body of information on successful implementation and use of small computer systems. It reviews the state of the art in current microcomputer technology including both hardware and software design as well as numerous applications of personal computing in education, recreation, business, etc. No computer expertise is required. Any individual who has a common sense understanding of computers can actively benefit from the course. The course will meet every Tuesday from 6 to 8 PM, from September 27 through October 18, and it costs $20. For further information contact Dean Patrick Basilice, Evening Division, St John's University, Jamaica NY 11439.■

Attention Computer Store Owners

WRC Smith Publishing Company will begin a new tabloid business paper to be known as *Computer Retailing*. A complete prototype was mailed to prospective subscribers in April 1977, with the first actual issue scheduled for July.

The publication will reach retail stores selling microcomputers to individuals for personal use and for use in small business. Some 300 such stores exist today. This number is expected to rapidly escalate as miniaturization and cost reductions bring computers within the reach of the average person. Contact Walter Mitchell Jr, WRC Smith Publishing Company, 1760 Peachtree Rd NW, Atlanta GA 30357.■

A New Toronto Store

A retail microcomputer store, Computer Mart, has opened at 1543 Bayview Av, Toronto, Ontario CANADA M1K 4K4. The store will feature computers for business and personal use. Systems will be available in assembled and kit form including spare parts and peripheral devices. The store will be stocked with tools and books, and demo systems will be up and running. Computer Mart will provide engineers and hobbyists the opportunity of putting their own systems together and will also offer inexpensive computing for small business.■

Processing Logical Expressions

(Or, how an ingenious extension of the Bauer-Samelson Algorithm allows us to compile efficient code for expressions involving "and," "or," "not" and relations among operands.)

W Douglas Maurer
University Library Rm 634
George Washington University
Washington DC 20052

Editor's Note: The first two articles by Dr Maurer appeared in BYTE's February and March 1976 issues. This article represents an extension of concepts presented in those articles, which are available in reprint form in the book Best of BYTE, Volume 1.

In a previous two part article, "Processing Algebraic Expressions," we presented the Bauer-Samelson algorithm and applied it to expressions involving the arithmetic operators and parentheses. Here we shall extend these ideas further and show how code can be generated in cases where relational operators ("greater than," "greater than or equal," and so on) and logical operators ("and," "or," "not" and so on) may be present.

Consider an IF statement (as in FORTRAN) such as:

IF (A-B .NE. C*D) GO TO 87

If we are compiling code for statements, the code we want to compile in this case, assuming we have a single register machine, would be:

1. Load A.
2. Subtract B.
3. Store in TEMP.
4. Load C.
5. Multiply by D.
6. Compare with TEMP.
7. On inequality, go to statement 87.

These last two steps are subject to a certain amount of variation depending on details of the computer involved. On machines with a condition code register (e.g., an 8080 or 6800), the steps will be as above. The comparison will set the condition code register, and the specific reference to the relation (.NE. in this case) that we are interested in is made only in step 7. On machines without a condition code register, but with a set of skip instructions, step 6 will be "skip the next instruction if the given register and TEMP are equal" and step 7 will be simply "jump to statement 87." If there are no skip instructions of this kind, then step 6 will be "subtract TEMP" and step 7 will be "if nonzero, jump to statement 87." In any case, we can represent these two steps by a single function, "if unequal to TEMP, go to statement 87." This function will always require two steps, as long as there can only be one address in each instruction, because we have to make reference to two addresses here, namely the address of TEMP and the address of statement 87.

Let us now consider the expression:

A–B .NE. C*D

as a logical expression. This means that it may include operators (such as .NE.) in addition to the standard arithmetic operators. It also means that it has the value "true" (if A–B is unequal to C*D) or "false" (otherwise). We would like to process this logical expression according to the Bauer-Samelson algorithm in order to generate code like that shown above. In the remainder of this article we will show how to do this.

What Precedence for Relational Operators?

Let us first consider precedence of operators. Recall that in typical algebraic expression processing, multiplication is defined as having a higher precedence than addition.

This means that in an expression such as:

W*X+Y*Z

we perform the multiplications first (W*X and Y*Z) before adding together the results. Because logical operations are defined as comparisons of values (e.g., the results of arithmetic operations) we must evaluate our arithmetic expressions first. Thus, it should be clear that any arithmetic operator (+,–,*, /) has higher precedence than any relational operator (.EQ., .NE., and so on). Thus in our expression:

A–B .NE. C*D

we have to perform the subtraction (A–B) and the multiplication (C*D) before comparing the results to see whether they are unequal.

There will never be any need to compare one relational operator with another with respect to precedence, and we can assume that all six relational operators (.EQ., .NE., .GT., .GE., .LT. and .LE.) have the same precedence. This allows us to start the process of generating code for our example expression. We recall that, in the Bauer-Samelson algorithm, we have two stacks, one for operands and one for operators (see figure 1). The process in this case proceeds as follows:

1. The A goes on the operand stack.
2. The minus sign goes on the operator stack.
3. The B goes on the operand stack.
4. Now we come to the .NE. which we cannot put on the operator stack because, as we mentioned before, the minus sign has higher precedence. Therefore, we must take the minus sign off the operator stack; take its operands, A and B, off the operand stack; perform the operation, which in this case means generating the code for "load A" and "subtract B"; and put the result back on the operand stack. In this case the result is simply the name of the register where the answer (A–B) was left. Let us call this register AC.
5. Now we can put .NE. on the operator stack, since the operator stack is empty.
6. The C goes on the operand stack.
7. The * goes on the operator stack, since it has higher precedence than the .NE. operator.
8. The D goes on the operand stack.
9. Now we come to the end of the expression. First we must treat the * operator (at the top of the operator stack). We take it off the operator stack; take its two operands (C and D) off the operand stack; and perform the operation, which in this case means generating the code for "load C" and "multiply by D." In doing this, we will note (if we have a single register computer) that the register AC will be overwritten. Therefore, we will have to generate "store in TEMP" and put TEMP on the operand stack where AC was. Finally, we put the result back on the operand stack. This time the result is the register where C*D is kept. Let us assume that this is again AC (although the method would work just as easily if it were some other register); then this is put back on the operand stack.
10. Finally, we come to the .NE. operator. Its two operands are AC and TEMP. Since we are only processing the expression A–B .NE. C*D, and not the whole statement in which it is contained, we don't know where to jump yet (in this case it is to statement number 87). Let us, therefore, generate code which will jump to a label called TRUE if the condition is satisfied, and to a label called FALSE if it is not. Later on,

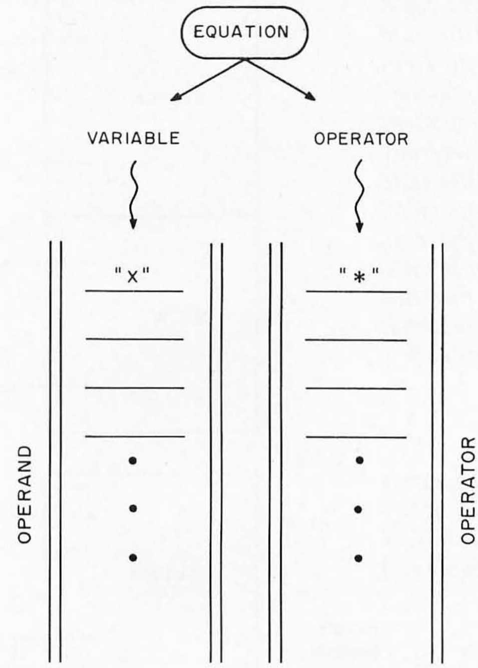


Figure 1: A simple diagram of the stack data constructs of the Bauer-Samelson algorithm. Take an equation and analyze it from left to right putting variables or constants in the operand stack and operators in the operator stack. With each additional item added to the stacks, the previous one is shifted down. Special attention must be paid to the operator stack because the operators must be stacked in descending precedence order. If an operator of lower precedence than the previous one is loaded into the operator stack, it will be saved, while the higher precedence operator and its corresponding operands are processed out of the stack. The newly computed value will be put into the operand stack, and the lower precedence operator will now be loaded into the operator stack. The process then continues on. When the end of the expression is reached, a cleanup process of unstacking operators and operands continues until both stacks are empty. This algorithm can be used in a compiler to generate machine code, or in an interpreter to control actual evaluations of expressions.

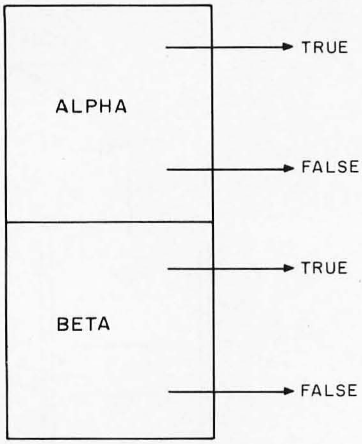


Figure 2a: The basic block diagram that will be used to represent the different logical operators; "alpha" and "beta" represent generated code whereby either a true or false state is possible, depending on whether or not the condition in question is satisfied. Arrows in this diagram indicate branches which might be taken.

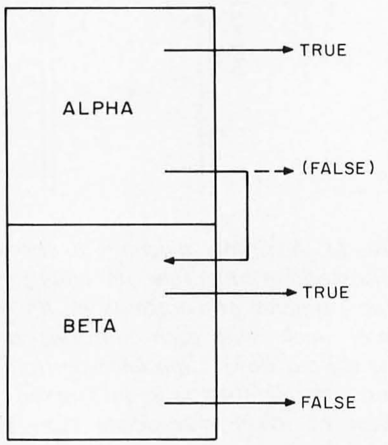


Figure 2b: Generating code for "alpha or beta." If the first state is not true then the second state must be checked:

Alpha		Beta	Result Branch
False	or	True	→ TRUE
False	or	False	→ FALSE
True	or	True*	→ TRUE
True	or	False*	→ TRUE

*State need not be checked.

these can be patched up: TRUE can be changed to statement number 87, and FALSE can be changed to be the next statement after the IF statement. So, in this case, depending on the computer we are doing this on, we would generate:

> "compare with TEMP; if unequal, go to TRUE; go to FALSE"
> or "if equal to TEMP, skip the next instruction; go to TRUE; go to FALSE"
> or "subtract TEMP; if nonzero, go to TRUE; go to FALSE."

We notice immediately that, no matter which of these options we take, the last instruction we generate is "go to FALSE," and this is unnecessary, because FALSE corresponds to the next instruction following the IF statement. In this case the final "go to FALSE" can simply be thrown away; however, in a more complex case we might actually need the "go to FALSE." So for the moment, let's assume that it is always generated.

Logical Operators

We now pass to a consideration of the logical operators "and," "or," and so on. Just as before, we start by considering questions of precedence. Suppose we have a statement such as:

> IF (A–B .GT. C*D .OR. E/F .LT. G+H) GO TO 87

It is now clear that we have three levels of precedence. That is:

1. First we perform the arithmetic operations: A–B, C*D, E/F, and G+H.
2. Then we look at the relational operators (.GT. and .LT. in this case) and form the expressions A–B .GT. C*D and E/F .LT. G+H.
3. Finally, we look at the logical operator (.OR.).

From this it should be clear that, just as arithmetic operators have higher precedence than relational ones, relational operators have higher precedence than logical ones. Suppose that we have more than one logical operator. For example, suppose we have a statement:

> IF (A.LT.B .OR. C.EQ.D .AND. E.GT.F) GO TO 84

Do we consider the .AND. first? Or do we look at the operators from left to right, which would mean considering the .OR. first? The answer to this question is not a natural result of the way we define logical operations in the same way that it is for the arithmetic operations such as + or *. In algebra and arithmetic, when we look at W*X+Y*Z, we know that by the ordinary rules of algebra the multiplications are performed first. But, there are no "ordinary rules of algebra" for logical expressions. The only clue we have is that "or" and "and" are often called logical sum and logical product, respectively, and are sometimes even denoted by + and * in mathematical logic. This would suggest that the "and," or the logical product, be performed first; in fact, this is the rule that is generally adopted. The logical operator "not," when it is allowed, has higher precedence than "and" and "or," while "implies," when it is allowed, generally has lower precedence.

In order to see how to generate object code when "or" is present, let us look at figures 2a and b. In accordance with what we have just discussed, the code we are trying to generate for an expression of the form "alpha or beta" will have various jumps to TRUE and FALSE. We do not care how many jumps there are, and in a complex case there may be quite a few of these. We assume, however, that we have already generated code for "alpha" and also code for "beta," and that at this point we only wish to take care of the "or" operator. The code for "alpha" comes first, followed by the code for "beta"; and the code for "alpha"

will have its own jumps to TRUE and FALSE, as will the code for "beta." This means that there are four possibilities:

1. A jump to TRUE from "alpha." If "alpha" is true, then clearly "alpha or beta" is true. So this can remain a jump to TRUE.
2. A jump to FALSE from "alpha." If "alpha" is false, we still have to check "beta." So this is changed into a jump to the start of the coding for "beta."
3. A jump to TRUE from "beta." If "beta" is true, then again "alpha or beta" is true, so this can remain a jump to TRUE.
4. A jump to FALSE from "beta." We don't check "beta" unless "alpha" is false, and we now know that both "alpha" and "beta" are false, so that "alpha or beta" is also false. So this can remain a jump to FALSE.

From this we can see that the only jumps we have to change (i.e., patch) are jumps to FALSE from "alpha." Each one of these must be changed from a jump to FALSE to a jump to the start of the coding for "beta" (see figure 2b). We have shown only one jump in the figure; but, if there is more than one such jump, all of them must be changed.

Now let us look at "and" (see figure 3). The setup is exactly the same as before; we still have four possibilities; but the rules are a bit different. Taking the four possibilities one by one, we consider:

1. A jump to TRUE from "alpha." If "alpha" is true, we still have to check "beta." So this is changed into a jump to the start of the coding for "beta."
2. A jump to FALSE from "alpha." If "alpha" is false, then "alpha and beta" is false. So this can remain a jump to FALSE.
3. A jump to TRUE from "beta." By this time we know that "alpha" is true, and if "beta" is also true, then "alpha and beta" is true. So this can remain a jump to TRUE.
4. A jump to FALSE from "beta." If "beta" is false, then again "alpha and beta" is false, so this remains a jump to FALSE.

Thus, it is only the jumps to TRUE from "alpha" (rather than the jumps to FALSE) which have to be changed to go to the start of the coding for "beta," as in figure 3.

In all cases of this kind, the coding for "beta" will immediately follow the coding for "alpha." If A is the start of "alpha" and B is the start of "beta," we only need to look through from A to B-1 in order to find jumps to FALSE (or, respectively, jumps to TRUE) that have to be changed to be jumps to B. The start of each section of generated code for an expression is what goes on the operand stack to represent that expression.

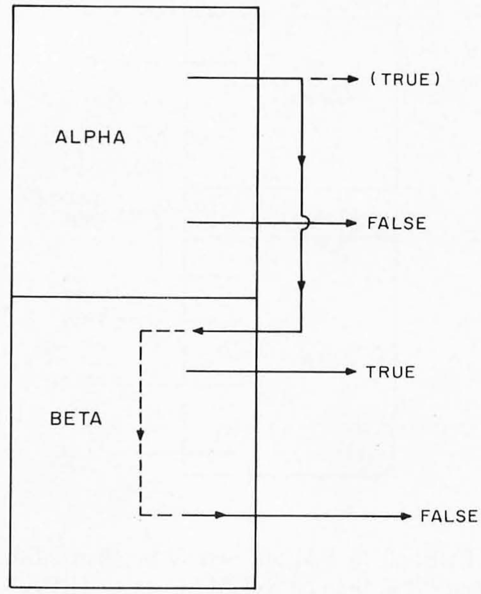


Figure 3: Generating code for "alpha and beta." With the "and" logical operator, both states must always be checked. If either state is false, the condition isn't satisfied and thus the end state is false.

Condition			Result
Alpha		Beta	Branch
True	and	True	TRUE
False	and	True*	FALSE
True	and	False	FALSE
False	and	False	FALSE

*State need not be checked.

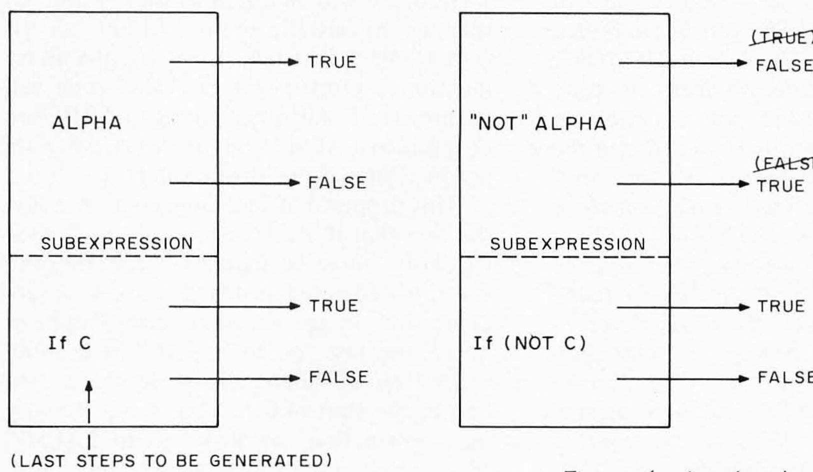


Figure 4: Another logical operator is the "not alpha." It can be thought of as the inverse of an alpha, true is false and false is true. There is also a modification on the subexpression as explained in the text.

A variable CREL (current relation) is kept at all times to indicate the start of the section of code generated for the current relation (e.g., A.LT.B). When we finish generating code for a relation, CREL goes on the operand stack, and the current position in the generated code, plus one, becomes the new value of CREL. At the beginning, CREL is initialized to indicate the start of all generated code.

Let us now look at "not alpha" (see figure 4). This seems easy: All the jumps to TRUE become jumps to FALSE, and vice versa. There is one exception, however. For any logical expression or subexpression, the generated code will end with something of the form "if C then go to TRUE; go to FALSE," which we shall abbreviate as "if C then DECIDE." This is clearly true of the code for relations, and it remains true when blocks "alpha" and "beta" are combined by means of "and" or "or," because "beta" will not be patched up. When we use "not," however, it would seem that "if C then go to

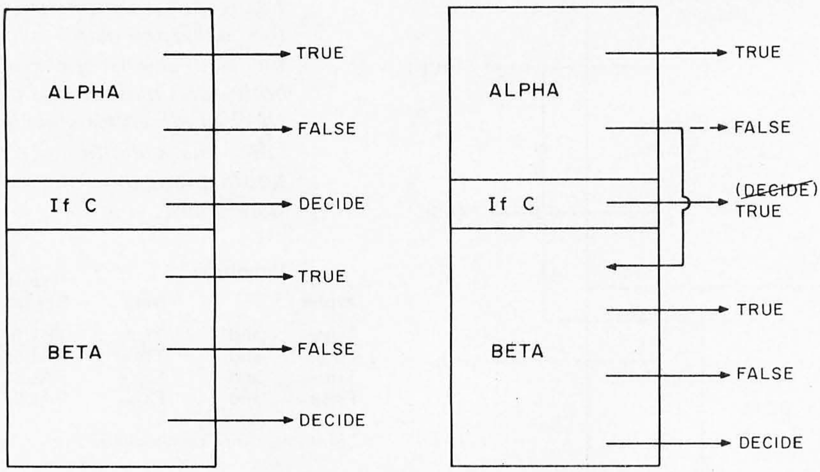


Figure 5: Optimized code for "alpha or beta." The subexpression "if C then go to TRUE; go to FALSE" can be simplified to "if C then DECIDE"; decide whether the code satisfies the situation in question or not. But, if the first state of the "or" operator isn't satisfied, then "go to FALSE" in the subexpression must be changed to "go to beta" which is the next step in the procedure. In the optimized form, the "go to FALSE" has been deleted and the subexpression has been simplified to "if C then go to TRUE."

TRUE; go to FALSE" would be changed to read "if C then go to FALSE; go to TRUE." Instead, we will change it to read "if (not C) then go to TRUE; go to FALSE" (or "if (not C) then DECIDE"), so that the above mentioned property of generated code will be preserved. All other jumps to TRUE are changed to FALSE, and vice versa, when we produce "not alpha" from "alpha."

This property of generated code—namely, the fact that it always ends with "if C, then DECIDE"—may be used to "optimize out" the last generated instruction for any subexpression. In generating code for "alpha or beta," the last "go to FALSE" in the subexpression of "alpha" must be changed to "go to the start of beta"; but this is the very next instruction, so that "go to FALSE" may simply be thrown away. Thus "if C then DECIDE" is changed, in this case, to "if C then go to TRUE" (see figure 5). In generating code for "alpha and beta," we can think of "if C then DECIDE" in the subexpression of "alpha" as if it were "if (not C) then go to FALSE; go to TRUE." This time "go to TRUE" must be patched so that it goes to the very next instruction, and it may likewise be thrown away. Therefore, "if C then DECIDE" may be changed in this case to "if (not C) then go to FALSE."

1. Load A.
2. Compare with B.
3. If less than B, DECIDE.
4. Load C.
5. Add D.
6. Compare with E.
7. If unequal to E, DECIDE.
8. Load G.
9. Multiply by H.
10. Compare with F.
11. If greater than or equal to F, DECIDE.
12. Continue with next instruction.

.
.
.

Figure 6: The steps involved in evaluating the FORTRAN expression:
*IF (A.LT.B .OR. C+D.EQ.E .AND. .NOT. F.GT.G*H) GO TO 87*
as compiled using the Bauer-Samelson algorithm.

We illustrate all these notions by generating code for:

IF (A.LT.B .OR. C+D.EQ.E .AND. .NOT. F.GT.G*H) GO TO 87

The generated code is given in figure 6; it is generated as follows:

1. CREL is initialized to indicate step 1.
2. A goes on the operand stack.
3. .LT. goes on the operator stack.
4. B goes on the operand stack.
5. .OR. has lower precedence than .LT., so we cannot put .OR. on the operator stack yet. We remove .LT. from the operator stack, and we remove its two operands, A and B, from the operand stack. We generate code for A.LT.B, namely "Load A; compare with B; if less than B, DECIDE," and put the result back on the operand stack. This result is the current value of CREL, namely "step 1," and CREL is reset to "step 4" since three steps have been generated so far.
6. Now we can put .OR. on the operator stack, since that stack is empty.
7. C goes on the operand stack.
8. + goes on the operator stack, since it has higher precedence than the .OR. operator.
9. D goes on the operand stack.
10. We cannot put .EQ. on the operator stack, because it has lower precedence than +. We take + off the operator stack, and its two operands, C and D, off the operand stack. We generate code for C+D, namely "load C; add D," and put the result, AC, back on the operand stack.
11. Now we can put .EQ. on the operator stack because it has higher precedence than .OR., which is at the top of that stack.
12. E goes on the operand stack.
13. We cannot put .AND. on the operator stack, because .EQ. has higher precedence. So we take .EQ. off the operator stack, and its arguments, namely AC and E, off the operand stack. The code we generate to compare AC with E is simply "compare with E; if equal to E, DECIDE." CREL, namely "step 4," goes on the operand stack, and since we have generated four steps since then, the next step would be number 8; so "step 8" becomes the new value of CREL.
14. Now we can put .AND. on the operator stack, because its precedence is higher than that of the .OR. operator.
15. .NOT. goes on the operator stack, since .AND. has lower precedence.
16. F goes on the operand stack.
17. .GT. goes on the operator stack, since .NOT. has lower precedence.
18. G goes on the operand stack.
19. * goes on the operator stack, since .GT. has lower precedence. The operator

stack now contains: .OR. (at the bottom); .AND., .NOT., .GT., and *.

20. H goes on the operand stack, which now contains "step 1," "step 4," F, G and H. We are now at the end of the expression.

21. First we treat the *. We take it off the operator stack; we take its two operands, G and H, off the operand stack; we generate code for G*H, namely "load G; multiply by H"; and we put the result, namely AC, back on the operand stack.

22. Now we treat the .GT. operator. Its two operands are F and AC, not AC and F. The result is the same as if .LT. were the operator and AC and F were the operands. (If F is greater than AC, then AC is less than F, not less than or equal.) We generate "compare with F; if less than F, DECIDE" and put CREL, namely "step 8," on the operand stack. As before, we have generated four new steps, so that "step 12" becomes the new value of CREL. Note that we have now generated all the steps of the compiled code shown in figure 6. The operand stack now contains "step 1," "step 4," and "step 8."

23. The next operator to be treated is the .NOT. operator. Its operand is "step 8," and "step 12" is the current value of CREL. This means that steps 8 through 11 have to be checked. In these steps, there are no jumps to TRUE or to FALSE (if there were, then TRUE would be changed to FALSE in these steps, and vice versa), but there is a reference to DECIDE in step 11. This step therefore becomes "if greater than or equal to F, DECIDE" and "step 8" remains on the operand stack.

24. Now we treat the .AND. operator. Its two operands are "step 4" (alpha) and "step 8" (beta), which means that we have to check from step 4 to step 7. Again there are no jumps to TRUE or FALSE. (A jump to TRUE would have to be changed, here, to a jump to step 8.) But there is a reference to DECIDE in step 7, which is changed to "if unequal to E, go to FALSE," and "step 4" goes back on the operand stack, since this is the beginning of the code for "alpha and beta."

25. The last operator to be treated is the .OR. operator. Its two operands are "step 1" (alpha) and "step 4" (beta), which means that we have to check from step 1 through step 3. If there were any jumps to FALSE in this range, they would have to be changed to "jump to step 4" (see figure 5). But, just as before, there are none of these; there is only the reference to DECIDE in step 3, and, again by figure 4, this is changed to "if less than B, go to TRUE."

26. Finally, we read the "GO TO 87," so that TRUE becomes statement number 87. There are now two jumps to TRUE (in steps 3 and 11), and these must be changed to refer to statement number 87; there is one jump to FALSE (in step 7), and this must be changed to refer to the next step after the generated code, namely step 12. This completes the process of generating code for this statement.

As a final note, we may consider the "implies" operator of symbolic logic. The construction "alpha implies beta" is equivalent to "(not alpha) or beta." Thus "implies" may be put on the operator stack, just like any of our other operators. It is treated in the same way as "or" (see figures 2 and 5), except that FALSE and TRUE are interchanged in "alpha" just beforehand (see figure 4).■

Comments on Notation

In this article, the logical and relational operators are shown using a FORTRAN IV convention, as follows. They are used to define conditions with relational operations, then combine them with logical operations.

Logical Operators: *A new condition is evaluated using other conditions X and Y.*

.NOT. X	*Logical inverse of condition X.*
X .AND. Y	*Logical product of results of conditions X and Y* $(X \wedge Y)$.
X .OR. Y	*Logical sum of conditions X and Y* $(X \vee Y)$.

Relational Operators: *A condition is evaluated using operands which can be compared, such as integers (or other representations of numbers), strings, etc.*

A .GT. B	*True if A is greater than B.*
A .GE.B	*True if A is greater than or equal to B.*
A .LT. B	*True if A is less than B.*
A .LE. B	*True if A is less than or equal to B.*
A .EQ. B	*True if A is equal to B.*
A .NE. B	*True if A is not equal to B.*

Note that the meaning of the comparisons depends upon the type of data involved, and that compiled code for each kind of test must reflect the comparison of identical types of data. Thus, A and B should both be integers in a simple compiler or interpreter (such as Tiny BASIC), and both should be of consistent integer or floating point (REAL) type in a FORTRAN compiler or interpreter, etc. The meaning of string comparisons for value (other than equality) can be defined.

Continued from page 12

attention to a few misconceptions. Referring to Prof Isaacson's article in February 1977 BYTE, I'll agree that the time of personal computing *is* coming, but it hasn't got here quite yet!

She may think that $1000 is cheap, and it is to some, but not for the vast majority of students and others. This is my main sticking point: too many writers assume one has $1000 to $2000 to blow on a system. Surprise! A lot of people don't! Professor Isaacson may think $1000 is nothing, but to me it is! Pardon me if I am old fashioned, but I like to get the most for my money!

I'd like to see more articles like the one on calculator interface, and Suding's article on TV typewriters. But why limit this to peripherals? Why not a homebrew computer that can be expanded and use BASIC? There are plenty of guys like me who enjoy "rolling your own." Not to mention, it's cheaper!

Edgar N Cohen
3228 Smith Av
Baltimore MD 21208

Homebrewing is definitely a learning experience, but hardly recommended as an economic alternative. The cost of individual parts may look small, but any realistic computer uses a very large number of the little bugs and inordinate amounts of time getting it up and running. The reasons for homebrewing are most likely psychic: the intimate knowledge and the chance to "do it right" by some arcane standards of system design only known to the builder of the system.

MICROCOMPUTER CORRESPONDENCE COURSES NEEDED . . .

I am trying to locate a formal course of instruction in microcomputer technology. I have considerable experience in third generation computer systems (maintenance, operations and programming at all levels). I also have worked with programmable calculators a great deal. Now my goal for both personal and professional interests seems to lie within the micro/mini computer range.

Unfortunately, my financial resources are somewhat limited, and the course of study must be limited to spare time with no restrictions on movement. The answer would be a correspondence course, preferrably VA approved.

If possible, I would like a course with kit construction as part of the curriculum. Not only would I then benefit by practical application of the knowledge being taught, but would retain the system for continued training and satisfaction, in addition to possibly being able to have some of the costs borne by the government. This would make it possible for me to continue my education in a manner that is currently beyond my means.

If you know of such a course, or might be influential in organizing such a course, then I would appreciate your help and influence.

Donald R Darden
519 Corday St, Lot 3
Pensacola FL 32503

Perhaps publishing your letter will have the influence you seek.

LAMENTS

I lament your loss of linkage on the 8008 update in April 1977 BYTE. The laborious data dependent search was eventually successful. Thus forced to acknowledge the anecdotal nature of this issue caused my correlator to flash *Readers' Digest* in red letters. Back to basics and I'll continue to sing your praise.

Frank Evans
Corvallis OR 97330

BEWARE OF PUNS

William Stock, in his letter ("What's New?", April 1977 BYTE), mentioned the sloppy disk, which was first introduced by our company in late 1976, as though he had never heard of it, not to mention our company. The unfortunate fact that many others are in the same position prompted this letter (which I am trying to key in slowly, as I know many "compuniks" are used to 110 baud and therefore cannot read very fast).

MNewmonics is currently based in Lawrence KS. The company made its start by marketing a small microprocessor, introduced soon after the popular 3000 series came out. By coincidence, our model is the MN1500, a 1 bit (not bit slice, but truly 1 bit) machine with two powerful instructions: 0 = NOP; 1 = HALT/CRASH. Although the actual fabrication is proprietary, I can say that it is a combination of the p-type and n-type bipolar (PPP, NNN) TTL components noted in William's letter. The device is capable of such extremely fast operation that we have developed a special dovetailed clock for it, which allows four states per cycle. I won't divulge its secret, but it uses an in-house NLL (noise locked loop).

We, too, have a WOM (write only memory), in 19 and 23 pin DIP formats. These are nicely complemented by our 64 K*1 static RNMs (read never memories), though some enterprising soul will probably soon discover that the same thing can be done with a 7404. Next, we have a RAM which covers several speed and power ranges. As we are a small company, we couldn't afford to support three chips differing only in speed, so we put the high, medium and low speed devices all in one package, and control them with an automatic gear shift, an industry first! The latest cheap,

excuse me, chip, which we developed locally was our MK*N RMM (read maybe memory). Also, my (barely) capable VP, Mr John C Nunley, has recently closed a deal making us the sole supplier of a new thin-film technology memory device. At one point, he was so worried about the deal that we have dubbed the ultra thin film (10**–8 cm) chip the ANGST-ROM.

We are currently trying to get our corner on the market with the product William mentioned, something we feel is a breakthrough for the microcomputer industry: the sloppy diskette. In passing, our prototype was made from a square piece of coated mylar housed in a circular sheath, which when placed in our drive sat motionless. Rather than the motor turning the diskette and a small flywheel, the entire drive began rotating with a negative frequency (another application of our dovetailed clock). The mangled test rack and the plaster patch in the wall are testimony to the fact that the idea was indeed a breakthrough!

Also, we have in the works a device which was suggested to us by one of our company insultants, Mr Craig Paul. The new device will be a yet smaller version of the increasingly popular diskettes. As we intend it to be used with the upcoming "baby" (pico) processors, we have tentatively named it the "Bassinette Drive."

Glen E Monaghan
President
MNewmonics
GEM/pdp-15 Lawrence KS 66044

MORE COMMENTS ON AUDIO TAPE PROBLEMS

Having read your reply to the letter by Warren Bell in the April 1977 issue I feel the problem should be clarified so that computer people out there don't start tearing into their cassette recorders needlessly. As I interpret Mr Bell's problem, it is unrelated to the phase of the tape recorder output, and reversing the output jack leads as you suggest won't help. The cassette recorder interface I've built which is on a IO card inside my SWTPC 6800 is similar to the AC30 that Mr Bell mentioned. The incompatability he experienced sounds like the problem I had with a prototype of my board. The SWTPC AC30 has 2400 Hz as the frequency that is recorded when there is no data coming out of the serial IO. I suspect that his interface rests at the lower frequency of 1200 Hz. My first board had the same problem. I could record and play my own tapes but it took several hours to find out why I could not read the BASIC I bought from SWTPC. As Mr Bell observed, you must invert the data before it is modulated to invert the frequency. My interface doesn't care if I invert the cassette output since it is frequency dependent, not phase dependent.

I might add my objection to the Kansas City interface using the upper frequency as the resting point. As the cassette starts up on unmodulated 2400 Hz it must pass through the lower frequency to get up speed. I would have thought the lower frequency would be used so the interface would not put out a burst of garbage data each time it starts up. My software must ignore any data for about a second as the cassette gets up speed.

Leo Taylor
18 Ridge Ct W, Apt C21
West Haven CT 06516

The phase inversion fix, if needed, is done with patch cords, not inside recorders!

ANOTHER COMMENT ON DEFINING THE BYTE

Not only does the quotation from Donald E Knuth published in the Letters column of April 1977 BYTE not address the question of the origin of the word "byte," it is not even intended to be a general definition. Professor Knuth was describing the basic unit of information on a particular computer, the MIX 1009. The passage quoted is taken from Knuth's *The Art of Computer Programming*, Volume I. There Knuth describes the mythical MIX computer, the machine language of which is used to illustrate the algorithms in the book:

> *MIX has a peculiar property in that it is both binary and decimal at the same time. The programmer doesn't actually know whether he is programming a machine with base 2 or base 10 arithmetic. This is done so that algorithms written in MIX can be used on either type of machine with little change; and so that MIX can be easily simulated on either type of machine.*
>
> *An algorithm in MIX should work properly regardless of how big a byte is.*

For those who may be looking for a general definition of "byte" I propose the following: A byte is the smallest directly addressable area of a computer's storage; if the smallest such area is the same size as the accumulator (or a general register) then it is more properly called a word.

Thus, an IBM 360 has an 8 bit byte size (smallest addressable storage location) and a 32 bit word (size of a general register). Operating under the above definition, each storage location of an 8 bit microcomputer would more properly be called a word than a byte.

Bob Brown
Chief of Computer Systems
Medical Association of Georgia
938 Peachtree St NE
Atlanta GA 30309 ■

What's New?

A New Computer Package from Polymorphic Systems



The POLY 88 System Sixteen computer system has been announced by Polymorphic Systems, 460 Ward Dr, Santa Barbara CA 93111. The unit is a 16 K system featuring a high speed video display and alphanumeric keyboard. Cassette tapes are used for permanent program storage; a verify feature is included to insure correct loading of programs. A BASIC package is included which features formatting options and string manipulation. A program library is available to POLY 88 users. The price for the system as shown is $2250, assembled. ∎

Circle 604 on inquiry card.

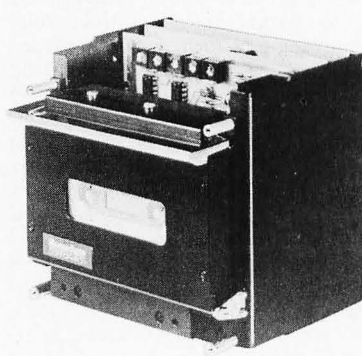
Parker Rolls into Microprocessor Games Market



Board game fans take note! Parker Brothers has announced a microprocessor-based board game called CODE NAME: SECTOR, a submarine pursuit game in which players compete with one another and with the computer as it moves an invisible submarine through 4800 possible positions under the sea. Parker Brothers expects that the price of this new product will probably be in the $30 to $45 range at department stores and other retail outlets this summer and fall. Significantly, there is no video screen involved with the game: a calculator-like module is used for both input and output, and apparently the entire game algorithm is built into read only memory parts with occasional inputs from players, through switches. For new players of the game, a unique "teach" mode is designed to give instruction regarding rules and operation. For more information, write Parker Brothers, Salem MA 01970.∎

Circle 605 on inquiry card.

A New Servocontrolled Cassette Transport

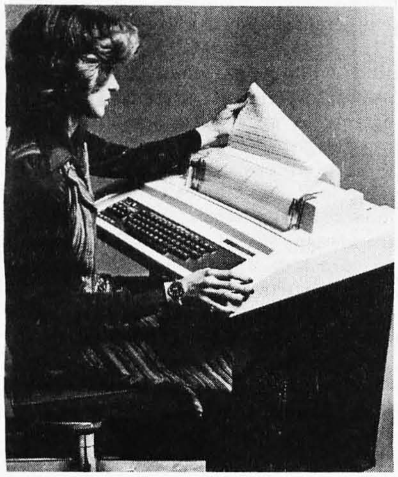


A new, patented, servodriven digital cassette recorder has been announced by Memodyne Corporation, 385 Elliot St, Newton Upper Falls MA 02164. The Model 764 consists of a tape transport attached to a card cage which contains the servocontrol electronics plus a read-while-write electronics card. Data can be recorded at up to 32 K bytes per second, and the transport search speed (forward or reverse) is 100 inches (254 cm) per second. (Thus a 300 foot

cassette has a maximum access time of 36 seconds. At 1600 flux changes per inch a standard cassette holds about 5 million bits, or about 1/2 million bytes. High quality certified digital cassettes should be used with this type of data density.) Start-stop settling time is 200 ms. The Model 764 is priced at under $500 in OEM quantities. Single unit prices were not listed on the data sheet we received. Contact Memodyne for more information.■

Circle 606 on inquiry card.

An APL-Compatible DECwriter



Digital Equipment Corporation has announced a new 1200 bps keyboard terminal called the LS120 DECwriter III, designed for interactive communication with a computer or as a unit in a data, or message, communication network. Features of the DECwriter III include an 180 cps print rate, a 1 K character buffer EIA interface, an automatic "paper out" switch, last character visibility and self-test capability. The new terminal is compatible with full duplex and half duplex protocols. Optional features include an APL alternate character set, a compressed font and a 300 bps acoustic coupler.

Price is $2270 in quantities of 100, with deliveries to begin this summer. Contact Digital Equipment Corporation, Digital Components Division, 1 Iron Way, Marlborough MA 01752.■

Circle 608 on inquiry card.

A New Terminal Kit from SwTPC

The Southwest Technical Products Corporation CT-64 Terminal System kit, along with the optional CT-VM video monitor, form a complete stand alone terminal system compatible with a variety of modems and ASCII computer systems. The kit features 16 lines of 32 or 64 characters per line, scrolling or page mode operation, upper and lower case characters, reversed character printing, control character printing, cursor control and complete control character decoding.

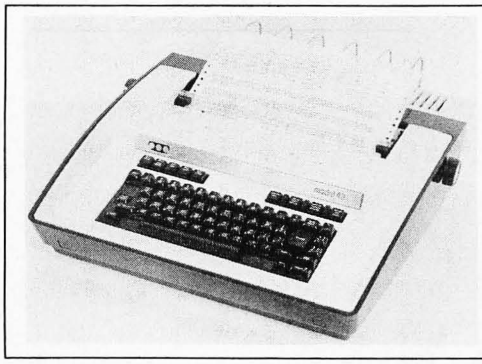
The unit is available as a kit only for $325. The price includes power supply, keyboard, serial interface, beeper, assembly instructions, chassis and cover. The optional CT-VM video monitor, sold assembled, requires the CT-64's power supply and sells for $175. Contact Southwest Technical Products Corporation, 219 W Rhapsody, San Antonio TX 78216.■



Circle 609 on inquiry card.

A Successor to the Teletype Model 33

Teletype Corporation, 5555 Touhy Av, Skokie IL 60076, has announced a new under $1000 Model 43 matrix teleprinter offering 30 character per second throughput, upper or lower case printing, and 132 column format capability on 11 inch wide fanfold paper. It features a 9 wire impact printhead mechanism. MOS technology is used throughout. The desk top terminal, which weighs approximately 30 pounds, is designed to be portable. Quiet operation makes it suitable for a variety of office environments. The 43 teleprinter is designed for interactive timesharing, data inquiry and response applications, and is interchangable with the Teletype Model 33 terminal. It is available with an EIA or 20 mA current loop interface and is designed for use on the switched net-



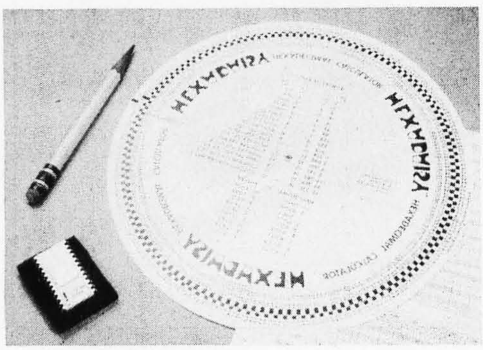
work and point-to-point private line facilities. The terminal includes operator controls for 10 or 30 characters per second, half or full duplex operation, parity checking and printer test. The terminal is comprised of five major pluggable components and features built-in self-diagnostics for troubleshooting. For further information call Tom Race (312) 982-3134.■

A sample of Teletype Model 43 listings:

```
Teletype Corporation
5555 Touhy Avenue
Skokie, Ill. 60076
```

Circle 610 on inquiry card.

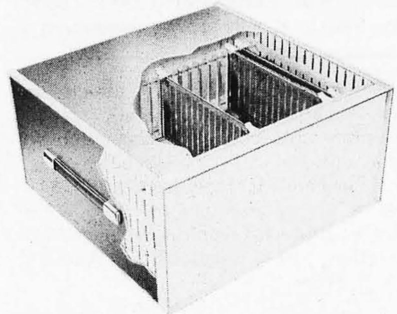
Circular Reasoning



HEXADAISY™ is a new device which calculates the offset for relative addressing as used by the 6800, Z-80, 6502, SC/MP and F-8 microprocessors, adds and subtracts hexadecimal numbers, calculates 2's complements and converts decimal to hexadecimal numbers and back. All results within the range of 0 to 255 (decimal) can be read directly, while larger numbers are calculated two hexadecimal digits at a time. The device is made from plastic in the form of a circular "slide rule" with waterproof printing. Available for $3.95 postpaid, with instructions, from E & L Pfeiffer Computer Products, Box 2624, Sepulveda CA 91343, or from local computer stores.■

Circle 612 on inquiry card.

Vector Announces New Enclosures



VECTOR-PAK, a new line of enclosures from Vector Electronic Company Inc, 12460 Gladstone Av, Sylmar CA 91342, is designed to house laboratory equipment, instruments, development systems and computers. Internal structures such as optional card guides, brackets and straps permit installation of cards, modules and electronic components. The units are available in 17 sizes and ten colors. They are designed for the experimenter with a custom personal computer system, who wants a professional looking enclosure. Prices range from $65 to $95 without internal structures. Prices with the internal structures range from $125 to $185. Custom unit prices on request. Delivery is stock to four weeks.■

Circle 613 on inquiry card.

A New Microcomputer High Level Language Compiler

PennMicro announces the development of a true, code emitting compiler for Intel microcomputer systems: the SLAM compiler. SLAM (Symbolic Language Adapted for Microcomputers) is a high level language which uses signed 16 bit decimal integers and simple English-like statements to program microcomputers without any need to learn assembly or machine language. SLAM is said to feature a variety of conditional and subroutine commands, plus signed decimal numbers, simplified print commands, symbolic variables, and English-like program statements.

The compiler is source code compatible with the previously announced SLAM interpreter. This concept allows programs to be created, modified and debugged using the interactive nature of the interpreter, then translated into hexadecimal 8080 object code in Intel standard format when the user's program design has stabilized. The compiler provides for parameter linkage conforming to Intel conventions, absolute memory pointers and machine language subroutine calls. SLAM occupies less than 8 K of RAM. Versions are available for the Intellec 8/Mod 80, MDS-800, or SBC80 systems. The compiler and interpreter are available on paper tape in Intel standard hexadecimal format. Ininstruction manuals are supplied with each. Price for either product is $99; both are available as a package for $175. Delivery is stock to two weeks. Customers should specify MDS, Intellec 8, or SBC system-type, and available user programmable memory (RAM) size.

For more information, contact PennMicro, POB 5073, Lancaster PA 17604.■

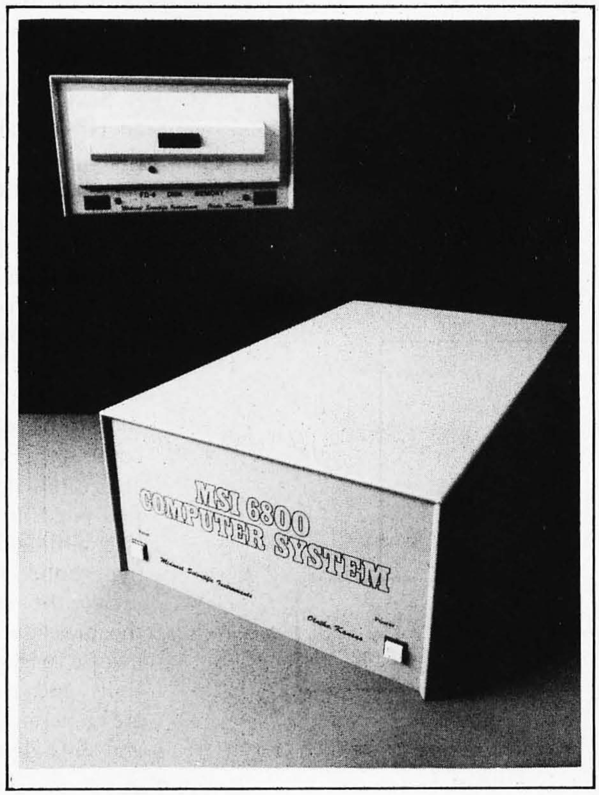
Circle 614 on inquiry card.

Building a better computer wasn't easy. But we did it.

Introducing the MSI 6800 Computer System

When we set out to build the new MSI 6800 Computer System, we knew we had our work cut out for us. It had to be at least as good as the now famous MSI FD-8 Floppy Disk Memory System which is also pictured below. So, the first thing we did was analyze all the problems and drawbacks we had encountered with other 6800 systems, and then put our engineers to work on solutions. The objective: Build a better computer.

We started with power supply. We had big ideas, so we used a hefty 18 amp power supply. You can run full memory and several peripherals without the worry of running out of juice. We also put it in the front of the cabinet so it's out of the way.



The next step was the CPU Board. A separate baud rate generator with strappable clock outputs allows any combination of baud rates up to 9600. A separate strappable system clock is available and allows CPU speeds of up to 2 MHz. The new MSI monitor is MIKBUG software compatible, so you will never have a problem with programs. Additional PROM sockets are available for your own special routines and to expand the monitor. The CPU also contains a single step capability for debugging software.

When we got to the Mother Board, we really made progress. It has 14 slots to give you plenty of room to expand your system to full memory capability, and is compatible with SS-50 bus architecture. Heavy duty bus lines are low impedance, low noise, and provide trouble-free operation.

With all this power and potential, the interface had to be something special. So instead of an interface address in the middle of memory, we put it at the top . . . which gives you a full 56K of continuous memory. Interfaces are strappable so they may be placed at any address. An interface adapter board is compatible with all existing SS-50 circuit boards and interface cards. All MSI interface cards communicate with the rear panel via a short ribbon cable which terminates with a DB-25 connector. All baud rate selection and other strappable options are brought to the connector so they may be automatically selected by whatever plug is inserted into the appropriate interface connector. Straps may also be installed on the circuit board.

To complete the system, we used an MSI 8K Memory Board which employs low power 2102 RAM memory chips and is configured to allow battery back-up power capability. A DIP switch unit allows quick selection of a starting address of the board at any 8K increment of memory.

If you're one of those people who understands the technical stuff, by now you'll agree the MSI 6800 is a better computer. If you're one who does not understand it yet, you'll be more interested in what the system can do . . . play games, conduct research and educational projects, control lab instruments, business applications, or just about anything else you might dream up that a microcomputer can do. The point is . . . the MSI 6800 will do it better.

The MSI 6800 Computer System is available in either kit form or wired and tested. Either way, you get a cabinet, power supply, CPU board, Mother board, Interface board, Memory board, documentation, instructions, schematics, and a programming manual. Everything you need.

There is more to say about the MSI 6800 than space permits. We suggest you send for more information which includes our free catalog of microcomputer products.

Building a better computer was not easy. Becoming the number one seller will be.

See the MSI 6800 Computer System at Personal Computing '77 - Atlantic City.

Midwest Scientific Instruments

220 West Cedar • Olathe, Kansas 66061
913/764-3273 • TWX 910 749 6403 (MSI OLAT)

Midwest Scientific Instruments
220 W. Cedar, Olathe, Kansas 66061

NAME ____________________

ADDRESS ____________________

CITY ____________________

STATE ____________________ ZIP ________

0010877

Simple Algorithms for Calculating Elementary Functions

John Rheinstein
10 Gould Rd
Lexington MA 02173

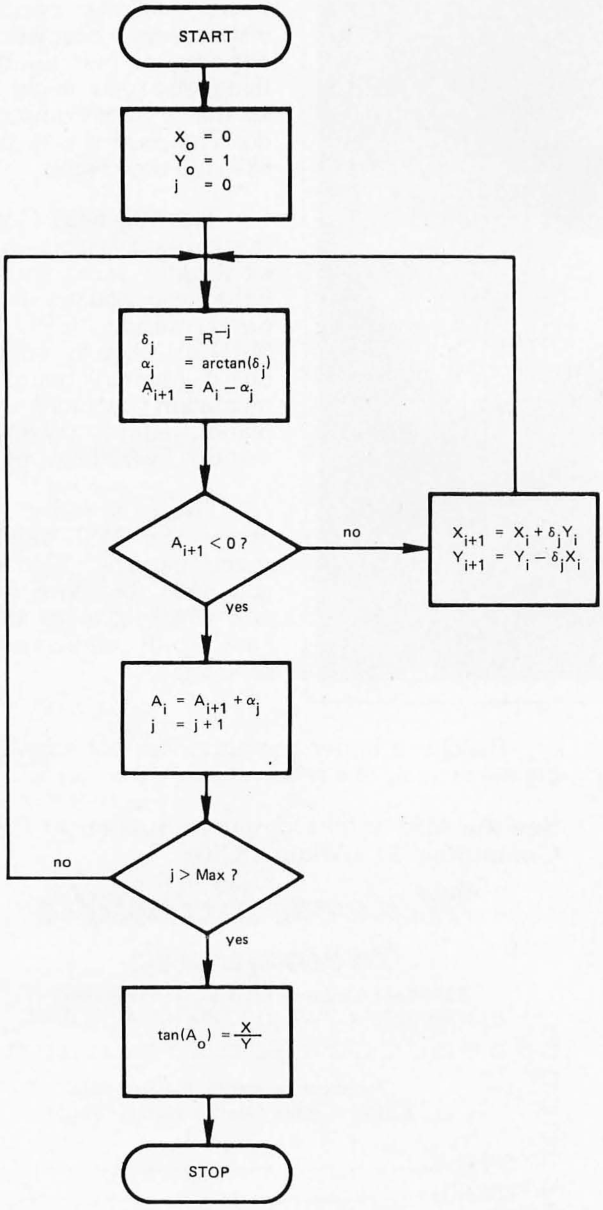


Figure 1: Flowcharted algorithm for calculating the tangent of A. It is preferable that the value of A be in the first circle, $0° \leqslant A \leqslant 360°$.

Several years ago, shortly after obtaining an HP-35 calculator, I became interested in the algorithm utilized by Hewlett-Packard to calculate the sine and cosine functions. Upon looking through the literature I found that the simple elegant algorithm employed is based upon the CORDIC technique developed by Volder. The basic algorithm was first described by Henry Briggs in 1624 in *Arithmetica Logarithmica*. This algorithm, which does not seem to be widely known, should be easily implemented in a microcomputer. Further, a simple variation of the algorithm, described by Walther in "A Unified Algorithm for Elementary Functions" (see bibliography), allows the calculation of hyperbolic functions and, more interestingly, the exponential function. Inverting the procedure allows the calculation of the inverse trigonometric and hyperbolic functions and the logarithm. These algorithms are generalizations of the well known add and shift algorithm used for multiplication.

Trigonometric Functions

The flowchart for calculating the tangent of an angle is given in figure 1. The algorithm may be carried out in any radix, R, generally the decimal (R = 10) or the binary (R = 2). The radix should be chosen so that the multiplication indicated in the iterative calculation of X_{i+1} and Y_{i+1} may be performed by shift operations only. Thus no multiplication or division is required until the final step of the algorithm is reached. The calculation may be in either degrees or radians, as long as either one is used consistently. α_j and δ_j should be prestored in memory and called as a look up table. It is not required that the calculation be performed in the first circle ($0° \leqslant A_o \leqslant 360°$); however, this is generally to be preferred. Care must be exercised if the final value of Y is very close to zero (A_o is close to 90°

or 270°) since division by a very small number, or zero, is apt to blow up. A separate test for this condition should be included. Max indicates the number of iterations through the major loop and should be at least one larger than the accuracy desired. For example, if an accuracy of about one part in R^{-5} is desired, then Max should be about 6.

The final value of X obtained after exiting the major loop is proportional to the sine of A_o, and the final value of Y is proportional to the cosine of A_o, the proper signs being maintained. Thus having obtained the tangent of A_o, the sine and cosine may be obtained.

$$\sin(A_o) = \frac{(\mathrm{sgn}X)(\tan(A_o))}{\sqrt{1 + \tan^2(A_o)}}$$

$$\cos(A_o) = \frac{\sin(A_o)}{\tan(A_o)}$$

where sgnX is the algebraic sign of X.

Hyperbolic Functions

A strikingly similar algorithm may be used to calculate the hyperbolic tangent of a number. For most of us this does not seem to be a very useful function, and the need for calculating it may not be clear. Having $\tanh(A_o)$ we can calculate $\sinh(A_o)$ and $\cosh(A_o)$.

$$\sinh(A_o) = \frac{\tanh(A_o)}{\sqrt{(1 - \tanh^2(A_o))}}$$

$$\cosh(A_o) = \frac{1}{\sqrt{(1 - \tanh^2(A_o))}}$$

Having the capability to calculate the hyperbolic sine and cosine is something that most of us could get along without.

The next step provides the exponential function, which is often of interest.

$$\exp(A_o) = \sinh(A_o) + \cosh(A_o)$$

Using the identities

$$\sinh(-x) = -\sinh(x)$$
$$\cosh(-x) = \cosh(x)$$

we also obtain

$$\exp(-A_o) = -\sinh(A_o) + \cosh(A_o).$$

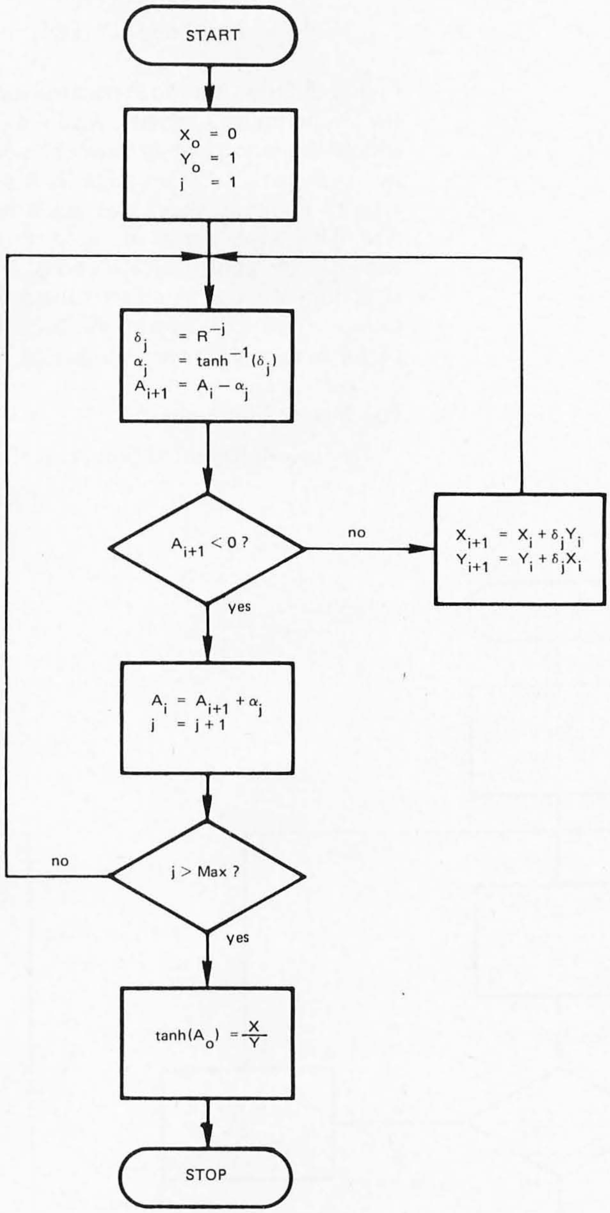


Figure 2: Flowchart for calculating the hyperbolic tangent of A. The value of A should be between 0 and 5 because the hyperbolic tangent rapidly approaches unity for values greater than 5.

The hyperbolic tangent approaches unity rapidly for arguments greater than about 5, causing a loss of accuracy in the algorithm. Thus it is best to prescale A_o to the range $0 \leqslant A_o \leqslant 1$. For prescaling, the following identity may be used.

$$\exp(Q\log_e(R) + D) = R^Q\exp(D).$$

Powers to any base may now be calculated. Setting D to zero above gives

$$\exp(Q\log_e(R)) = R^Q$$

from which we can obtain, for example

$$10^Q = \exp(2.3025851Q)$$
$$2^Q = \exp(0.69314718Q).$$

Figure 2 shows the flowchart for calculating the hyperbolic tangent. Again α_j and δ_j should be prestored in memory and called by means of a look up table. It is relatively easy to combine these two algorithms into one. This saves some memory, but at the expense of control instructions. For this algorithm we again find that the multiplications carried out within the loop may be performed by shift operations only.

The Inverse Functions

Having implemented these two algorithms utilizing table look up procedures for the αs and δs, the inverse tangent and inverse hyperbolic tangent can easily be calculated with the same look up tables. Flowcharts are shown in figures 3 and 4. The arcsine and arccosine as well as the inverse hyperbolic sine and cosine may also be easily obtained. Another function of more general interest, the logarithm, is obtained from the inverse hyperbolic tangent ($\tanh^{-1}$) by making use of the identity

$$\log_e(x) = 2\tanh^{-1}\left(\frac{x-1}{x+1}\right).$$

The logarithm to any base, b, may be obtained

$$\log_b(x) = \log_e(x)/(\log_e(b)).$$

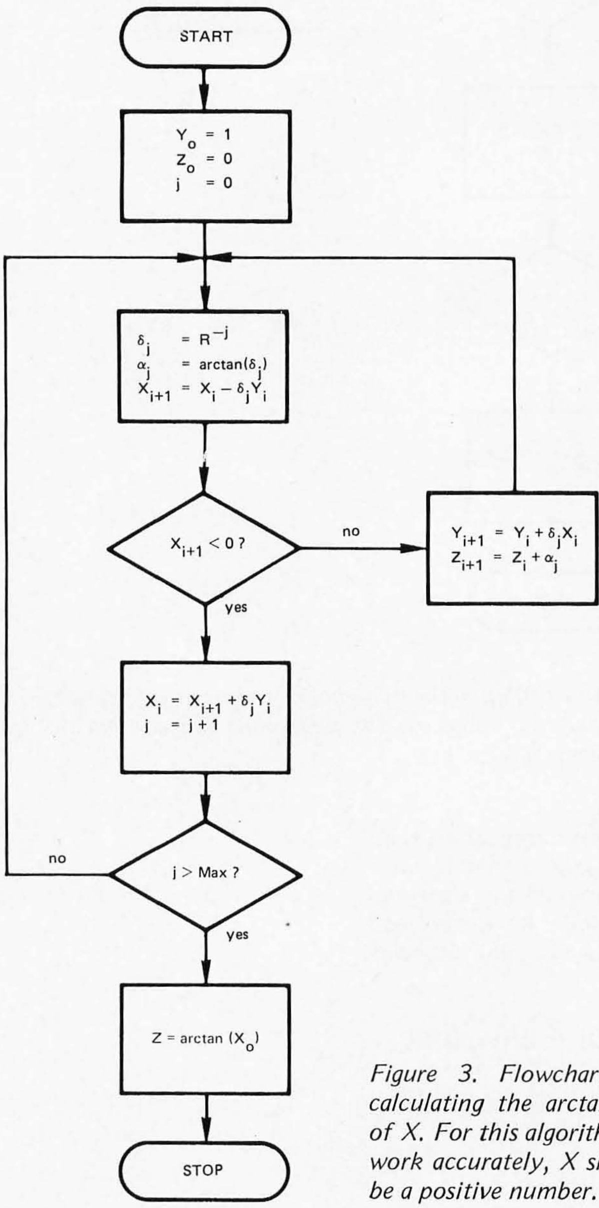


Figure 3. Flowchart for calculating the arctangent of X. For this algorithm to work accurately, X should be a positive number.

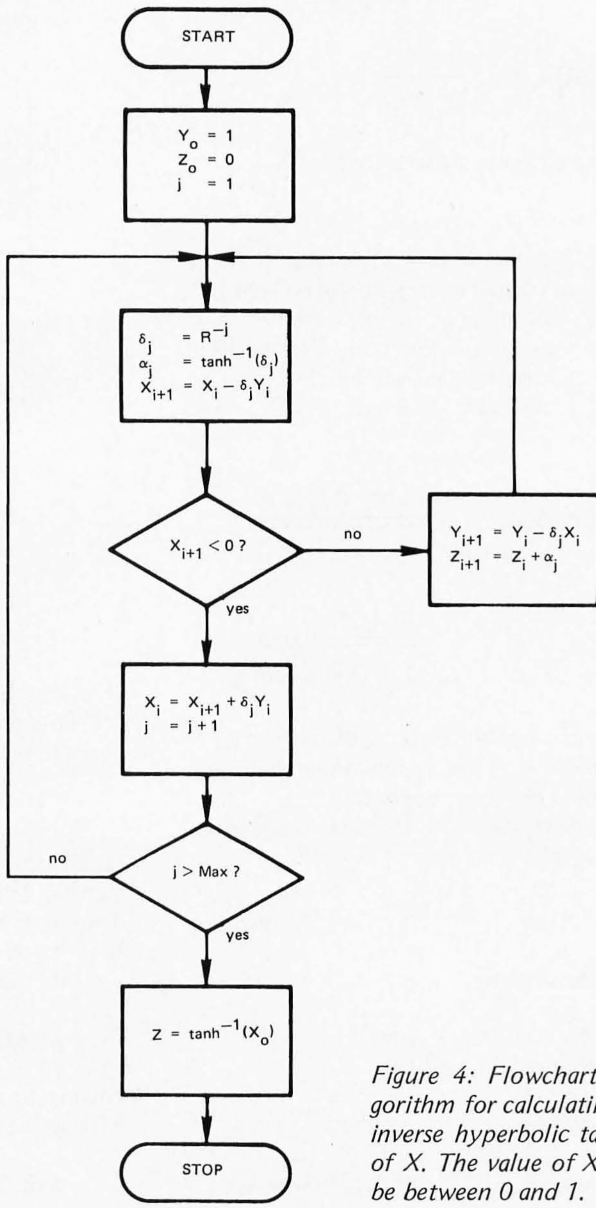


Figure 4: Flowcharted algorithm for calculating the inverse hyperbolic tangent of X. The value of X must be between 0 and 1.

When calculating logarithms, prescaling should be carried out so that only the mantissa is calculated.

Given the arctangent, the arcsine and arccosine may be calculated from

$$\arcsin(x) = \arctan\left(\frac{x}{\sqrt{1-x^2}}\right)$$

$$\arccos(x) = \arctan\left(\frac{\sqrt{1-x^2}}{x}\right)$$

If the argument of the arccosine is negative, 180° should be added to the result obtained from the above equation.

Given the inverse hyperbolic tangent, or logarithm, the inverse hyperbolic sine and cosine may be found.

$$\sinh^{-1}(x) = \log_e\left(x + \sqrt{(1+x^2)}\right)$$

$$\cosh^{-1}(x) = \log_e\left(x + \sqrt{(x^2-1)}\right)$$

Basic Program

A listing of a BASIC program which uses these algorithms to calculate the tangent and arctangent is shown in listing 1 to indicate the manner in which these algorithms can be used. The program was written in MaxiBASIC for the Digital Group Z-80 system but it should run on almost any BASIC with only minor modifications, if any. This program operates in radix 10 (R = 10). I also tested a version in radix 2 (R = 2) changing only statements 10 and 20 and the DATA statements.

This program will operate well in BASIC as listed, but the real power of the algorithms can best be obtained if a machine language version is utilized. The multiplication indicated in statements 220, 230, 380 and 400 consume much of the time. In a machine language version these may be replaced by shift operations which take considerably less time.

Comments

These algorithms could easily be used to augment the set of available functions written in BASIC as subroutines or they could be added as machine language subroutines for high speed operation. The accuracy can be made quite high. For the sample program seven significant digits will be obtained.

I do not know whether Hewlett-Packard uses algorithms similar to these for all of the functions described. Walther reports that Hewlett-Packard did construct a hardware floating point processor of a generalized version of these algorithms, and reports excellent performance.■

Listing 1: A BASIC program for determining the tangent or arctangent of an angle. The program was written using the flowcharts of figures 1 and 3. The sample run gives a demonstration of the accuracy attained with these algorithms.

```
5   REM** PROGRAM TO CALCULATE TAN OR ARCTAN OF
6   REM AN ANGLE (IN DEGREES)

10  LET R = 10
20  LET M = 8
30  FOR J = 0 TO M
35  REM** D(J) = DELTA SUB J
40  LET D(J) = R↑(–J)
45  REM** A1(J) = ALPHA SUB J
50  READ A1(J)
60  NEXT J
70  PRINT "DO YOU WANT TAN (ENTER 1) OR ARCTAN (ENTER 0)";
80  INPUT B
90  IF B = 1 THEN 120
100 IF B = 0 THEN 300
110 GOTO 70
120 PRINT "INPUT A(0)";
130 INPUT A
140 LET W = A
150 LET X = 0
160 LET Y = 1
170 FOR J = 0 TO M
180 LET A = A -- A1(J)
190 IF A <0 THEN 250
200 LET X1 = X
210 LET Y1 = Y
220 LET Y = Y1 – D(J)*X1
230 LET X = X1 + D(J)*Y1
240 GOTO 180
250 LET A = A + A1(J)
260 NEXT J
270 LET T = X/Y
280 PRINT "TAN(";W;") = ";T
290 GOTO 70
300 PRINT "INPUT X(0)>0";
310 INPUT X
320 LET W = X
330 LET Y = 1
340 LET Z = 0
350 FOR J = 0 TO M
360 LET X1 = X
370 LET Y1 = Y
380 LET X = X1 – D(J)*Y1
390 IF X < 0 THEN 430
400 LET Y = Y1 + D(J)*X1
410 LET Z = Z + A1(J)
420 GOTO 360
430 LET X = X1
440 NEXT J
450 PRINT "ARCTAN(";W;") = ";Z
460 GOTO 70
470 DATA 45, 5.7105931, 5.729387E-1, 5.729576E-2, 5.7295779E-3
480 DATA 5.729578E-4, 5.729578E-5, 5.729578E-6, 5.729578E-7
999 END
```

Sample Runs:

```
DO YOU WANT TAN (ENTER1) OR ARCTAN (ENTER 0)? 1
INPUT A(0)? 26.3
TAN(26.3) = .49423074
DO YOU WANT TAN (ENTER1) OR ARCTAN (ENTER 0)? 0
INPUT X(0) > 0? .49423074
TAN( .49423074) = 26.300001
ARCTAN (.49423074) = 26.300001
```

1. Jack E Volder, "The CORDIC Trigonometric Computing Technique," *IRE Transactions on Electronic Computers,* September 1959, page 330.

2. J S Walther, "A Unified Algorithm for Elementary Functions," *Proceeding, Spring Joint Computer Conference,* 1971, page 379.

Clubs and Newsletters

Unofficial Cosmac Users' Newsletter

We recently obtained firm evidence of a group of strange beings in a remote corner of New York State. Their latest communique read very much like this:

There exist small groups of computer buffs who are not interested in buying a complete "off the shelf" computer system, or even a kit. These strange people would rather take a single microprocessor chip, study it, play with it and caress the documentation until they have learned about possible hardware options. These misfits in the world of printed circuit boards then dig out their wire wrap tools (which they use so that their wiring errors are only semipermanent) and have the audacity to build their first systems for less than the price of a good tape recorder. True, these systems are very limited, but they grow easily as cash and ideas become available, until they rival any of the ready-made systems at a fraction of the cost.

One such group of people, who happen to have chosen the RCA CDP 1802 as the basic building block for their systems, exists in Western New York.

Their interest in this chip was kindled by the Popular Electronics *article on the Cosmac Elf. They have added hexadecimal keyboards, control systems, el-cheapo 6 digit displays and expanded memory to the tiny Elf. Some of these systems are capable of timing an egg, displaying the time of day, playing games and other vitally important things.*

All letters containing worthwhile, workable hardware or software ideas for the CDP 1802, and a stamped self-addressed envelope will entitle their authors to receive a free copy of the Unofficial Cosmac Users Newsletter.

Future editions of this dubious technical publication may or may not be available, depending on response.

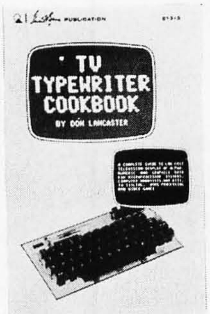
Terry Wolfe, *Cosmac Newsletter*
3812 Orchard St
Walworth NY 14568.

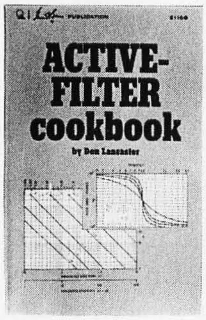
The West Virginia Computer Society...

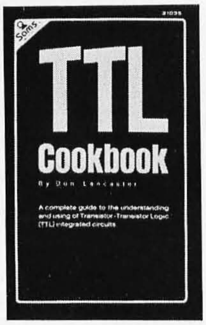
...is off to a very fast start. Their first

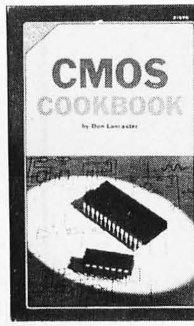
What's Cooking?

. . . The Lancaster Series!









_____**TV Typewriter Cookbook** by Don Lancaster. A complete guide to low cost television display of alphanumeric data, several chapters of which were published ahead of the book in early issues of BYTE magazine. $9.95.

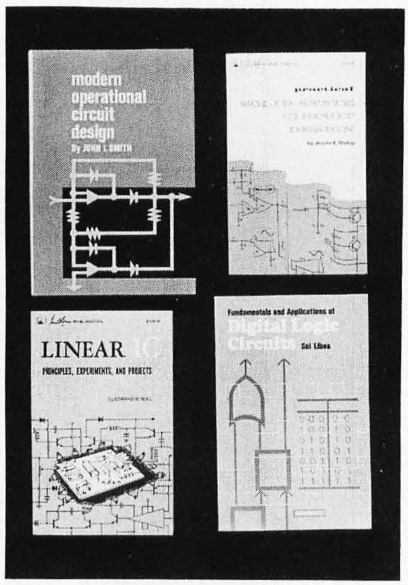
_____**Active Filter Cookbook** by Don Lancaster. The chief chef of electronics Cookbooks concocts another gourmet appetizer. Run to this book when you need to find a starting point for the design of a filter for use in an electronic application. $14.95.

_____**The TTL Cookbook by Don Lancaster.** Start here with Don's tutorial explanations of what makes TTL logic design tick. 335 pages, $8.95.

_____**CMOS Cookbook.** Don Lancaster has taken his highly successful TTL Cookbook formula and done it again. This bigger than ever (414 pages!) book tells you how to use CMOS, the exciting new power logic family. $9.95.

_____**Modern Operational Circuit Design** by John L Smith. An absolutely essential introduction to the use and application of operational amplifier systems. The book contains both theoretical background information and practical circuit suggestions which can be used to advantage by the experimenter. $23.25 (hardbound).

_____**Linear IC Principles, Experiments, and Projects** by Edward M Noll. From basic principles to complicated systems, from simple amplifier experiments to applications in radio, TV and control systems, this book can improve your knowledge of the way circuitry of the analog world really works. $8.95.



_____**Practical Solid-State Circuit Design** by Jerome E Oleksy. A self study course in the design of semiconductor circuits from the simple transistor to the complex operational amplifier. $5.95.

_____**Fundamentals and Applications of Digital Logic Circuits** by Sol Libes. An invaluable tutorial background volume on digital logic, arithmetic, IO concepts and interfacing to analog devices; written by one of the founders of the Amateur Computer Group of NJ. This book acquaints the reader with much of the terminology and background concepts of digital hardware. $6.95.

Send to:

BITS, Inc.
70 Main Street
Peterborough NH 03458

Check Payment method:

_____My check is enclosed
_____Bill my MC No.____________________Exp. date __________
_____Bill my BAC No.____________________Exp. date __________

Name ______________________________
Address ______________________________
City __________ State ______ Zip Code ________
Signature ______________________________

Total for all books checked $______
Postage 50 cents per book for ______ books $______
Grand Total $______

You may photocopy this page if you wish to leave your BYTE intact.

Prices shown are subject to change without notice.
All orders must be prepaid.
In unusual cases, processing may exceed 30 days.

Circle 121 on inquiry card.

meeting drew nearly 70 hobbyists. A newsletter is already off the press just a few weeks after their initial get together.

New clubs naturally need active imaginative people to form a strong nucleus. So, West Virginians, here's a chance to get in on the ground floor of what could well be a successful venture. Computer oriented ham and CB people are more than welcome. Bill England is the man to contact at 167 Iroquois Trl, Ona WV 25545, (304) 736-9794.

Southern New England Computer Society

SNECS is a well-established group serving Connecticut and Western Massachusetts. Meeting places vary but the Hartford area is most often chosen. For more information write Southern New England Computer Society, 267 Willow St, New Haven CT 06511, or call (203) 562-4739.

The Eastern Iowa Computer Club

Hobbyists may contact Samuel E Dillon, 1125 Washington Dr, Marion OH 52302, for information regarding the Eastern Iowa Computer Club.

Amateur Computer Group of NJ

ACGNJ continues to grow steadily in membership and ability to provide club services. Group purchase opportunities, workshops and a consistently improving newsletter are among this club's accomplishments. Included in the May issue of the *ACGNJ News* are a series on "Selectric Interfacing" by Charles Yates, a "6800 Disassembler" test program by T D Farnsworth, a "6800 Memory Test Program" by Brian Loofbourrow and an article on "Interfacing a Papertape Reader to the SwTPC 6800" by Michael P Burton. Write Amateur Computer Group of NJ, UCTI, 1776 Raritan Rd, Scotch Plains NJ 07076.

Ventura County Computer Society

Write POB 525, Port Hueneme CA 93041, for information about the Ventura County Computer Society.

KIM Users' Notes – A Change of Address

Eric Rehnke, Publisher of *KIM Users' Notes* has changed his address from 425 Meadow Ln, Seven Hills OH 44131 to 1330 Summitt Blvd, Broadview OH 44147.

Japan Microcomputer Club

RAMS' *Memory Pages* printed a letter from Koji Yada, a member of the Japan Microcomputer Club. Mr Yada says that the

club has grown to a membership of over 1000 since its formation last Fall. They would be glad to establish a newsletter exchange. Anyone speak Japanese? Write Koji Yada, Manager, Computer Center, Electrotechnical Laboratory, Tanashi Branch, 5-4-1 Mukodi-Machi, Tanashi-Shi, Tokyo JAPAN.

Charlotte Computer Society

The Charlotte Computer Society meets on the third Wednesday in Room 200, Benny Hall, at the University of NC at Charlotte. No address or phone number was given.

The Computer Network of Kansas City

Those hobbyists looking for club activity in the Kansas City area should get in touch with the Computer Network of Kansas City, 968 Kansas Av, Kansas City KN 66105.

Rochester NY – RAMS

There's something refreshing about *Memory Pages*, the publication of the Rochester Area Microcomputer Society. It's not a large or impressive newsletter by most standards, but the articles tend to be clear and intriguing. Hats off to a small club that does its work well. One new series that holds some promise is "How Not to Build a Microcomputer System" by Peter Helmers, a BYTE contributor. Here's a chance to cash in on an expert's mistakes. The series started recently and should run for some time. The address is RAMS, POB D, Rochester NY 14609.

Permian Basin Group in Texas

The Permian Basin Computer Group in the Midland and Odessa area of Texas is still meeting once a month. The Midland Group meets the second Monday of each month at 7:30 PM in the Student Union Building on the Midland College campus. The Odessa Group meets the second Saturday of each month at 1 PM in the Electronic Technology Building, Room 203, on the Odessa College campus. For additional information write John Rabenaldt, POB 3912, Odessa TX 79760, or phone (915) 332-9151, 9 AM to 5 PM Monday thru Friday or (915) 697-4607, after 6 PM.

Are You Looking for a Way to Give Your Club Greater Visibility?

Just send a letter to Peter Travisano, BYTE Publications Inc, Peterborough NH 03458. If you put out a newsletter, send that along, too, or put us on your regular mailing list. ■

What's New?

Altair (S-100) Bus Interface New Products

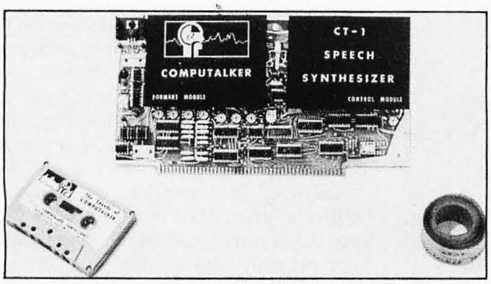
If one thing is fairly obvious, it is that the Altair 8800 bus, sometimes referred to in advertisements as the S-100 bus, is de facto a standard among many of the products we've seen to date in this industry of personal computing systems. This point was made quite dramatically when we went through the new products file for this month and found 12 different releases about various products which interface to the Altair bus. The products range widely in purpose and intent, but we've grouped them together since they address a common backplane definition.■

Speech Synthesizer

Computalker Consultants (D Lloyd Rice et al) have sent this picture of the Model CT-1 Speech Synthesizer, a $395 assembled and tested module for the Altair (S-100) bus. The Model CT-1 is designed to optimize the trade-off between low data rate speech and directly digitized speech. Low data rate speech relies on canned definitions for the sound of each phoneme, which produces mechanical sounding speech. Digitized speech, while remaining faithful to the original sound, requires 10 K to 20 K bytes per second of storage and is inflexible to phonetic manipulation.

With the Computalker Model CT-1, the sounds are defined in real time under software control. Parameters which represent the phonetic structure of human speech are transmitted to the CT-1 at a rate of 500 to 900 bytes per second, depending on the data compression techniques used. This allows the production of highly intelligible and quite natural sounding speech output. Speaker characteristics and language or dialect variations are retained in the output.

The Computalker Model CT-1 can also be operated in a low data rate mode using phoneme definitions contained in the CSR1 Synthesis-by-Rule software package, which is available for $35. The Computalker speech synthesis system, used in this way, has the



advantage that the software driver can easily be modified to keep the naturalness and intelligibility of the speech output up to date with the constantly evolving state of the art of rule governed speech.

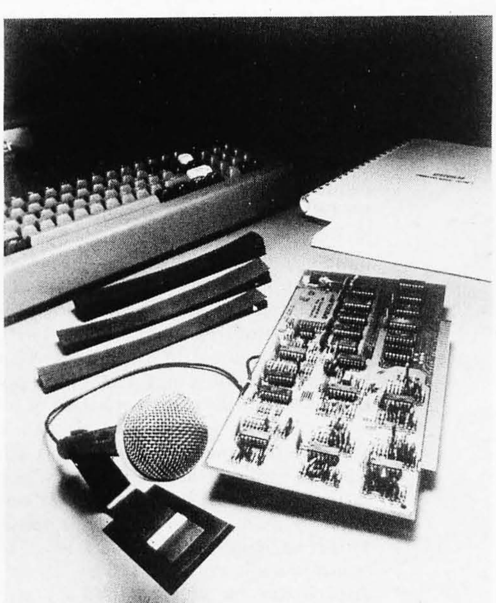
The Model CT-1 is a factory assembled and tested board, 5 1/4 by 10 inches, which occupies a single card slot. It requires a block of 16 output locations, one byte (8 bits) each, relocatable to any hexadecimal boundary via an onboard selector switch. Power requirements are +8 V unregulated (or +5 V regulated) at 170 mA typical, 250 mA maximum, and ±16 V unregulated (or ±12 V regulated) at 85 mA typical. Delivery is from stock to 45 days. Computalker Consultants is located at POB 1951, Santa Monica CA 90406.■

Circle 616 on inquiry card.

Vocal Input to your Computer

A new system compatible with all Altair (S-100) bus computers to provide voice input and control, and replacing keyboards in many instances, has been announced by Heuristics, a Los Altos CA speech research firm. Known as SpeechLab™, the system is available in selected computer stores and directly from the manufacturer for $249 in kit form or $229 assembled and tested.

SpeechLab digitizes and extracts data from speech waveforms and applies pattern matching techniques in order to



recognize vocal input. Applications include computer input, games, research and vocal control.

In addition to Altair bus computers, SpeechLab can be used with any computer by using a separate power supply and connector.

Included in the price is a complete hardware and software system, a 275 page laboratory manual, 95 page hardware manual, high fidelity microphone, and three programs on paper tape. The lab manual includes 35 graded experiments with over 100 tables and graphs.

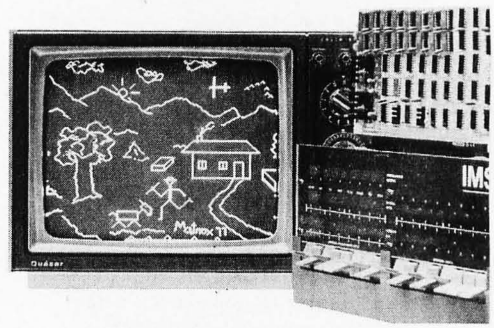
The recognition of spoken inputs requires 64 bytes of storage per word and a vocabulary of up to 64 words in memory. Other features include real time response, 95% recognition accuracy, automatic hardware self-test capability, and C-MOS design for low power dissipation.

Software includes the SpeechBasic™ BASIC programming language, assembly language speech recognition program, SpeechBasic plot, correlation, recognition, advanced recognition and hardware self-test programs.

For more information write to Heuristics Inc, 900 N San Antonio Rd, Los Altos CA 94022.■

Circle 617 on inquiry card.

A Modular Graphics Subsystem

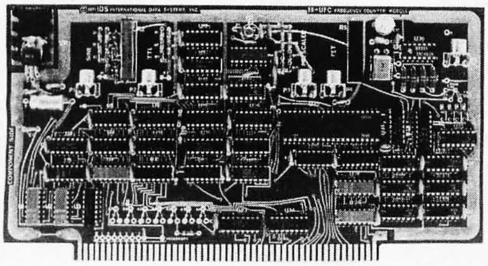


Matrox Electronic Systems, located at POB 56, Ahuntsic Station, Montreal, Quebec CANADA H3L 3N5, has sent along some information on what has come to be called the ALT-256**2 Graphics System in their product line of display devices. The ALT-256**2, to which we alluded in a previous new product story (see November 1976 BYTE, page 60) is designed to plug directly into any Altair (S-100) bus compatible computer such as the IMSAI shown in the press release photo. The board interfaces to the host computer as a set of four peripheral ports. Two of the ports are used for storing the X and Y coordinates of a spot, and a third port is used simply to turn the dot on or off; a fourth port is dedicated to clearing or presetting the whole screen. It contains its own 65,536 bit refresh memory so that no address space in the host computer is used up for display purposes. This is an important consideration, since multiple copies of this 256 by 256 grid display can be plugged into the same computer and jumpered to use common

synchronization so that a color effect with 256 by 256 resolution and seven colors can be achieved by driving a commercial RGB monitor (red-green-blue guns on separate inputs with common synch). The board is not a kit, and is priced at $395 FOB Montreal, with delivery two to four weeks after receipt of orders. European readers will also appreciate the fact that the ALT-256**2 is also available in a standard version using the European television timing definitions.■

Circle 618 on inquiry card.

Instrumentation and Test with the Personal Computer Bus



International Data Systems has announced an instrumentation module for the Altair (S-100) bus which allows the user to make measurements of frequency and period, formerly done with "dumb" uncomputerized measuring equipment. The product is the 88-UFC, a 9 decade universal frequency counter module. The design allows software selection of all functions and has capabilities said to have been previously available only on counters costing several times its $179 price. The 88-UFC has provision for four signal sources which can be selected under software control. Three of the inputs have general purpose analog or digital signal conditioning, and the fourth accepts TTL signals for use in debugging logic systems. One of the general purpose inputs includes a divide by ten prescalar which extends its input range typically above 600 MHz, and the other two general purpose inputs can count signals ranging typically up to 65 MHz. With one of these boards and the software provided, the user can in principle perform nearly all the frequency and time period measurements encountered in the typical electronics or scientific laboratory (to say nothing of the amateur radio station or electronics repair shop). The crystal controlled time base on the board can be selected in software to provide count periods from 100 ns to 1 second, with half cycle or full cycle counts. The board is said to allow measurement of a 600 MHz signal's frequency with ±1 Hz resolution, or two parts in 10^9.

Software included with the unit provides totalizing or accumulation of event counts, measurement of the period of a signal, measurement of frequency, computation of frequency ratios

Continued on page 162

BYTE's Bits

A Personal Computing Electronic News Service: the "Net" Worth of Space Age Journalism

We recently received a copy of the Washington Amateur Computer Society's *Journal* which contains news about the society, product reviews and speculative articles. In short, a very good example of the sort of journalism coming out of the local computer clubs.

What makes this particular issue so special, though, is that it is available via computer link to anyone possessing a computer terminal and telephone coupler! Members and nonmembers alike can dial (202) 635-5730 (for a 300 bps line) or (202) 635-5710 (for a 110 bps line) and start receiving the latest scoop from the Washington DC area.

The *Journal* is written and edited each month on a DECsystem-10 at the Catholic University of America; it is available as a "free access" text file (ie: no password or account number is necessary).

The procedure used to obtain the journal is simple. After calling either of the two numbers listed above, the DECsystem will respond with its answer tone. Activate your modem and type "Control C." The computer should echo an up-arrow followed by "C." (If it does not, repeat the procedure several times. If you still have no luck, the computer is probably down.) Next, type "I" followed by a carriage return. The system will reply with the following response (there may be some slight variations in this response): "CATHOLIC U. 507B20 19:39:01 TTY60 SYSTEM 95," followed by a period on the next line to indicate that you are conversing with the timesharing monitor and that it is waiting for your instructions. To get the *Journal*, type "HELP WACS" and a carriage return. The computer will then type out the journal text. The journal is set up to fit in a format of 60 characters to the line and 66 lines per page. If you are going to read the text on a video display unit, it is a good idea to tape the telephone output for later reference. To stop the output at any time, type "Control O."

It should be mentioned that the *Journal* is rather lengthy (16 pages for the April issue). The question of long distance phone charges for this type of service is not a trivial one. The following excerpts are taken from a letter sent to us by the directors of the club in which they discuss this and other points:

> *We believe that this is the first such electronic "mail service" or journal. You will not be working with a recording but rather will be running a program on the timesharing system that moves information stored on a disk file to a serial communications port. We would not be able to share this new adventure in communications with you if it were not for the kind assistance of the computer center of the Catholic University of America. Since this service uses computer time, which slows the computer down slightly and ties up a communications port, we ask that you use it only during nonbusiness hours: late at night or weekends would be ideal.*
>
> *Our journal is also available in printed form. . . . We hope that this will be the first of many such computer hobbyist communication nets, and that the age of electronic mail and news journals will not be far off. We understand that long distance telephone rates will make our service rather expensive for the individual, let's say, in California. But perhaps that is where hobbyist clubs can show their worth and make use of unused business WATS lines late at night or on weekends. . . . [Electronic journals] are just the kind of justification for the adoption of ASCII as a legal transmission code on the amateur airwaves that the FCC will recognize. We computer hobbyists need access to quick, efficient and up to date communications media to enable us to grow and exchange ideas.*
>
> *Why can't we have access to radio communications in the code that is a national standard not only for our hobby but also for ANSI? Computer hobbyists unite! Petition the FCC to allow us to communicate in our "native tongue." Meanwhile, find those fallow WATS lines and make them serve our hobby.*

Robert Jones, director
Thomas Crone, assistant director
Richard Bonanno, editor
William Stewart, assistant editor
Washington Amateur Computer Society
4201 Massachusetts Av #168
Washington DC 20016 ■

Pushover stacks are characteristic features of incompetent waiters.■

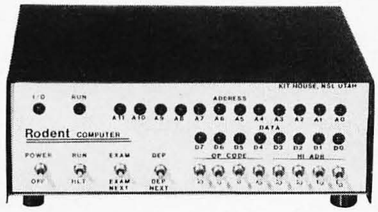
What is a strip of leather used to hold integrated circuits into the sockets of a voice analysis system?

A DIP thong ■

Attention KIM-1 and other 6502 system users. If you are tired of waiting for slow-slow program loads, we have the solution. A cassette write/read program that requires no additional interface hardware! Just two single-bit I/O ports! 213 bytes per second fast! Self-clocking with checksum and check read. Uses moderately priced recorders and cheap cassettes. $25.00 check or money order covers program and use instructions. Other copyrighted 6502 software available. Write for particulars. (California residents include 6% sales tax.)

INNOVATIVE SOFTWARE, INC.
3107 Casa Bonita Dr.
Bonita, CA 92002

Circle 149 on inquiry card.



PARALLEL I/O, POWER SUPPLY, 256 BYTE RAM, 16 OP CODES, TTL (Open Col) COMPATIBLE, CABINET.

Rodent

COMPUTER KIT $129.00
4K ROM/RAM kit (Less ROM)
3K RAM $99.00
KITS IN STOCK

KIT HOUSE
161 GARY WAY
NSL, UTAH 84054

Circle 174 on inquiry card.

IMAGINE

What you can do with a 4K RAM that can be positioned anywhere in memory by program commands.

We have some ideas you may not have thought of. Write and ask about our new 4K PL (program locatable) RAM card for the S100 bus.

The 4K PLRAM comes fully assembled, tested and burned in for only

U.S. PRICE	IN CANADA
$179.	**$199.**

Sorry - Blame Pierre
Ont. Residents
Add 7% PST.

E.M.A. **industries ltd.**
P.O. Box 421 Simcoe, Ontario, Canada N3Y 4L5

Circle 175 on inquiry card.

Disc/3

DISC/3 COMPANY

COMPLETE MICRO BUSINESS SYSTEMS

A/R, Mailing Labels, G/L, Payroll, A/P, etc.
plus

Lear Siegler ADM-3A kit	$ 749.95
IMSAI 8080 kit	$ 599.95
Centronics Business Printer (132 columns)	$1520.00

Disc/3

DISC/3 COMPANY
1840 LINCOLN BLVD.
SANTA MONICA, CALIF. 90404
(213) 451-8911

Circle 176 on inquiry card.

COMPUTER MART OF NEW HAMPSHIRE

DEALERS FOR

TDL ICOM IMSAI

DIGITAL GROUP

SOFTWARE INCLUDES:

8K BASIC EXT. BASIC
TEXT EDITOR DISK BASIC
WORD PROCESSOR
MACRO-ASSEMBLER

170 MAIN STREET NASHUA
(603) 883-2386

Circle 177 on inquiry card.

MICRO — BYTE
PRESENTS
A

PROGRAMMING TAPE CASSETTE

Designed Especially For Micro Computers

30 Minutes (15 minutes each side)
HIGH FREQUENCY
SUPER SENSITIVE
LOW NOISE

Florida Residents add 4% Sales Tax

$2.00 each

Micro - Byte
5870 MIAMI LAKES DRIVE EAST
MIAMI LAKES, FL 33014 (305) 822-6010

Circle 169 on inquiry card.

KIM-1 USERS

Are you in need of peripheral support to get the job done?

HERE ARE SOME OF THE THINGS YOU'VE BEEN WAITING FOR!

2704/2708 EPROM PROGRAMMER	$275
2K ROM BOARD with 2 1702A roms	$105
8K ROM BOARD with 2 2708 roms	$155
2K RAM BOARD with 16 2102-1 rams	$105
4K RAM BOARD with 32 2102-1 rams	$165
8K RAM BOARD with 64 2102-1 rams	$275
ONE PASS ASSEMBLER BOARD for KIM-1	$245
64 Char. ASCII TV INTERFACE BOARD	$255
8 In RELAY OUTPUT BOARD 120v, 400W	$195
MOTHERBOARD 6 card KIM-1 to S-100 bus	$95
MOTHERBOARD 6 card KIM-1 to EXORCZR bus	$95
MOTHERBOARD 6 card KIM-1 to KIM-1	$105
EXPANDER CABLE to couple motherboards	$45
KEYBOARD (63key)/EDITOR/ASSEMBLER with 8K RAM and TV INTERFACE	$895

All units fully assembled

*Pa. Res. add 6% Sales Tax

HYBRID TECHNOLOGIES
P.O. BOX 163
BURNHAM, PA 17009

Dealer Inquiry Welcome

Circle 179 on inquiry card.

Araeco



TPR-1

THE ULTIMATE IN:

OPTICAL TAPE READERS

Precision machined tape guide
Interface directly to a PIA or UART
Tri-state buffer output
All handshake logic · TTL compatable
Pull thru tape · Reads 0 TO 5000 cps
Interface thru a 14 pin IC plug (PROVIDED)

TPR-1	ASSEMBLED & TESTED	$38.00*
HSA-1	PLUG-IN HANDSHAKE ADAPTER	$ 5.50*

* POSTPAID - WIS. RESIDENTS ADD 4% SALES TAX

TO ORDER, SEND CHECK OR M.O. TO:

MICROCOMP P.O. BOX 1221 FOND DU LAC, WI 54935

WE SPECIALIZE IN MICROCOMPUTER SYSTEMS FOR
HOBBY - BUSSINESS - EDUCATION
WE ARE EXCLUSIVE **OSI** DEALERS...
--- WE DO A BETTER JOB.

-KITS- FULL SYSTEMS- PERIPHERALS- FLOPPY DISKS- ACCESSORIES-
-DIGITAL CLOCK KITS- VIDEO TERMINALS- BOOKS- MAGAZINES-
SEND FOR FREE CATALOG TODAY!!!

Circle 157 on inquiry card.

FOR YOUR ALTAIR OR OTHER S-100 BUS

UP TO 50% OFF

NEW FACTORY WIRED & TESTED BOARDS & SPECIAL COMPUTERS (mfg. by MITS)

- *12 BIT A/D*
- *2SIO*
- *CPU*
- *4K STATIC RAM*
- *24 INPUT MUX*
- *8K PROM BOARDS (W/W.O. PROMS) & OTHERS*
- *2708 PROMS*
- *ALSO NEW GE TERMINET 30s (110 or 300 BAUD)*

CALL DICK REEVES
(314) 569-1506

Circle 181 on inquiry card.

A Trapping Technique for the 8080

John M Schulein
1186 Arlington La
San Jose CA 95129

Debugging 8080 Software? How About a Trap Routine?

If you have got your 8080 system up and running and you are starting to write some programs for your system then the trap routine discussed in this article may be just what you need to assist you in debugging your own software or other software you have keyed in or relocated. This trap routine is in PROM in my system and has been invaluable in locating bugs in various programs.

This trap routine is a program that is written as a subroutine and is designed so that when it is called it will do the following:

1. Immediately save the machine state.
2. Print out the machine state on a television display.
3. Wait for a keyboard input so that the programmer can decide whether to examine or modify memory or to restore the machine state and resume execution of the program that called the trap routine.

A typical way to use this trap routine is to change one or more instructions in critical areas of the program being debugged to the 8080's RST 7 instruction (FF hexadecimal). Note that this implies that your problem program is in programmable memory and that if a multibyte instruction is changed, the RST 7 instruction is placed in the first byte of the multibyte instruction. Several traps can be set at once since the trap location will be printed out each time the trap routine is executed. The trap routine is activated (called) whenever the 8080 processor encounters an RST 7 instruction. The RST 7 instruction assumes there is a trap handler routine beginning at hexadecimal location 0038 in memory, so if the trap routine does not start at this location then a JMP TRAP instruction must be present instead. If your PROM memory starts at location 0000 then you can fix the trap routine (or jump instruction and trap routine) into PROM and you are ready to go. If your PROM is in high memory, then you can fix the trap rotuine in high memory and you will have to remember to load the jump to trap instruction (three bytes) into locations hexadecimal 0038 thru 003A each time you initialize your system. An alternate way to get into the trap routine would be to use a standard three byte call instruction. (Here no jump instruction would be required.) I decided to use the single byte call (RST) mainly for convenience (one byte to change instead of three) and also because my PROM is in low memory.

The output of my trap routine is two lines of listing on a television display as shown below:

```
TRAP AT :xxxx:
Axxxx:Bxxxx:Dxxxx:Hxxxx:Sxxxx:
|A|↑| |B|C| |D|E| |H|L| |SP|
  flags
```

Line	Label	Op Operand	Commentary
1	TRAP	PUSH PSW	
2		PUSH B	Save registers for final return to the program being debugged.
3		PUSH D	
4		PUSH H	
5		PUSH H	
6		PUSH PSW	
7		LXI H, 000EH	Put corrected SP onto stack and restore HL. Save and restore flags.
8		DAD SP	
9		POP PSW	
10		XTHL	
11		PUSH H	
12		PUSH D	
13		PUSH B	Save registers for display.
14		PUSH PSW	
15		LXI H, Trap Message	Display "TRAP AT"
16		CALL PRST	
17		LXI H,0013H	
18		DAD SP	
19		MOV D,M	
20		DCX H	Calculate trap location, result in HL.
21		MOV E,M	
22		DCX D	
23		XCHG	
24		CALL PTADR	Display trap location as four hexadecimal digits.
25		CALL CRLF	
26		MVI D,"A"	
27		CALL PRREG	
28		MVI D,"B"	
29		CALL PRREG	
30		MVI D,"D"	Display flags, registers, and SP.
31		CALL PRREG	
32		MVI D,"H"	
33		CALL PRREG	
34		MVI D,"S"	
35		CALL PRREG	
36		CALL RDTTY	Wait for keyboard input (ESC forces jump to monitor).
37		CALL CRLF	
38		POP H	
39		POP D	Restore machine state.
40		POP B	
41		POP PSW	
42		RET	Return to program being debugged.
43	PRREG	CALL PRNT	
44		POP H	
45		XTHL	Subroutine to POP a trap register through the top of the stack and then print the result.
46		CALL PTADR1	
47		RET	
48	TRAP-MESS	'TRAP AT∇'	

Listing 1: Symbolic assembly language listing of the trap routine used by the author in his 8080 system. Detailed comments by line number are made in the text of the article. Note that in the author's system, a hardware vector to the monitor occurs if an ESC key is pressed on his terminal keyboard instead of some other character when the program is looking for input at line 36. The system routine referenced by this program are summarized in table 1.

The first line shows the memory location of the RST 7 instruction that invoked the trap routine and the second line displays the machine state (accumulator, flags, the registers B,C,D,E,H and L, and the stack pointer) at the completion of the last program instruction executed before the trap calling instruction (RST 7) was encountered.

The trap routine is shown in listing 1 and the auxiliary subroutines used by the trap routine are described in table 1. Listings of these auxiliary subroutines are not provided since most monitor programs will already have these routines and they most likely are peculiar to individual systems since they deal directly with IO devices. The stack data generated by the trap routine is shown in figure 1 and will help in understanding the operation of the trap routine.

When the trap subroutine is called by a RST 7 instruction, the program counter is automatically pushed onto the stack. The first four push instructions in the trap routine (listing 1, lines 1 thru 4) save the A,B,C,D,E,H and L registers and the flags in the stack. This machine state information will be used at the very end of the trap routine to restore the machine state just before the trap routine return instruction (line 42 of listing 1) is executed. The fifth push instruction (line 5) is used to provide a space in the stack for the value of the stack pointer when the trap routine was called. This value is calculated by adding 000E hexadecimal to the current stack pointer value at line 7 of the trap routine. The DAD SP instruction (line 8) does this addition and the desired initial stack pointer value ends up in HL after the DAD SP instruction is executed. Note that the DAD SP instruction affects the carry flag, and thus the PUSH PSW and POP PSW instructions at lines 6

Table 1: Summary of Auxiliary Subroutines. This table lists the symbolic name referenced in listing 1 and the corresponding functions of several monitor routines required. No listings of these routines are provided since the detailed code will vary from system to system.

Symbol	Function
PRST	Display (as on a printer) a string of ASCII characters in memory starting at the location pointed to by HL. The string is terminated by a zero byte.
PTADR	Display (print) HL as four hexadecimal digits surrounded by colons.
PTADR1	Display (print) HL as four hexadecimal digits surrounded by colons.
CRLF	Sends a carriage return and a line feed (or equivalent) to the output device.
RDTTY	Reads an ASCII character from the keyboard. Any character other than ESC returns the processor to the calling program with the ASCII character in the A and D registers. An ESC character sends the processor to the monitor routine.
PRNT	Display (print) the ASCII character in the D register.

and 9 are required to preserve the flag byte.

The XTHL instruction at line 10 exchanges the most recent two stack bytes with the contents of HL. Thus after the execution of line 10, the initial value of the stack pointer is on the bottom of the stack (8080 stack grows downward) and the contents of HL have been restored to their initial values (the values they had when the trap routine was called). The next four push instructions (lines 11 thru 14) save the machine state a second time for the printing operation.

Lines 15 and 16 print the ASCII character string "TRAP AT" in preparation for the printing of the memory address that generated the trap. This information is highly desirable since several traps may have been set into the program being debugged and it is nice to know which trap was encountered each time the trap routine is executed. The trap subroutine return address (pushed onto the stack when the trap subroutine was called) is the key to determining the trap location. In order to access this information, line 17 and 18 cause the HL register pair to point higher up in the stack to the most significant byte of the return address. The return address is put into the DE register pair by the instructions at lines 19, 20 and 21. Note that the value of the DE register pair is decremented by one (line 22) in order to reflect precisely the location of the RST 7 trap generating instruction. This is due to the fact that the return address pushed onto the stack by the RST 7 instruction is the next memory address after the RST 7 instruction. The XCHG instruction at line 23 exchanges the register pairs DE and HL which puts the trap location into HL so that the PTADR subroutine called at line 24 will print the trap location. (PTADR prints out the contents of HL as four hexadecimal digits surrounded by colons.) A CRLF subroutine called at line 25 moves the output display device to the beginning of the next line.

Now things get a little tricky, but we are almost through so don't give up yet. The next ten lines of program (lines 26 thru 35) result in the printing out of the *machine* state in the form of five groups of characters on the display using the PRREG subroutine (lines 43 thru 47). Each group of characters consists of a single identifying alphabetic character, four hexadecimal digits, and a colon. Thus the printout of the machine state line takes 30 characters and just fits on a 32 character line.

Each execution of the PRREG subroutine results in the following operations. First the ASCII character in the D register is printed out on the display by the PRNT subroutine. Next the return address for the PRREG subroutine is popped off the stack into the HL register pair by the POP H instruction at line 44. This brings to the bottom of the stack one of the two byte values to be printed that was placed on the stack by the instructions at lines 10 thru 14. Now the PRREG subroutine return address is restored to the stack (two bytes up from its original position) and the two byte value at the bottom of the stack that is to be printed is put into HL by the XTHL instruction at line 45. (That XTHL instruction sure is nice!) Finally the four hexadecimal digit value now in HL is printed out on the display by subroutine PTADR1 (called at line 46). Each execution of the PRREG subroutine causes the stack to be reduced by two bytes and after the execution of the last PRREG call instruction (line 35) the stack has been reduced to the upper half of that shown in figure 1.

After the entire machine state line has been printed out, the trap routine waits for a keyboard input in the RDTTY subroutine (called at line 36). This gives the person debugging a program a chance to digest the trap information before proceeding. Any character other than ESC will return the processor to the trap routine and then to the program being debugged. An ESC will cause

the processor to return to the monitor program where the operator can examine and modify memory if necessary. If the character typed was not an ESC, a CRLF is generated and then the machine state is popped off the stack by the four pop instructions at line 38 thru 41. After restoring the machine state, the processor will now return to the program containing the trap (the program being debugged) and resume execution. Note that you can only use this technique if the program being debugged has the trap calling instructions (RST 7) located in program locations that originally contained NOPs. Most of the time I use the ESC monitor exit so that I can restore the original instruction and move the RST 7 trap calling instruction to a new location. Then another execution of the problem program can be initiated. This process will continue until the source of a problem is determined and then the program can be modified to (hopefully) correct the problem.

This trap routine (or a similar one) should make debugging much easier as it produces a printout of the entire machine state at a known point in a program. Good luck in finding and removing those bugs!■

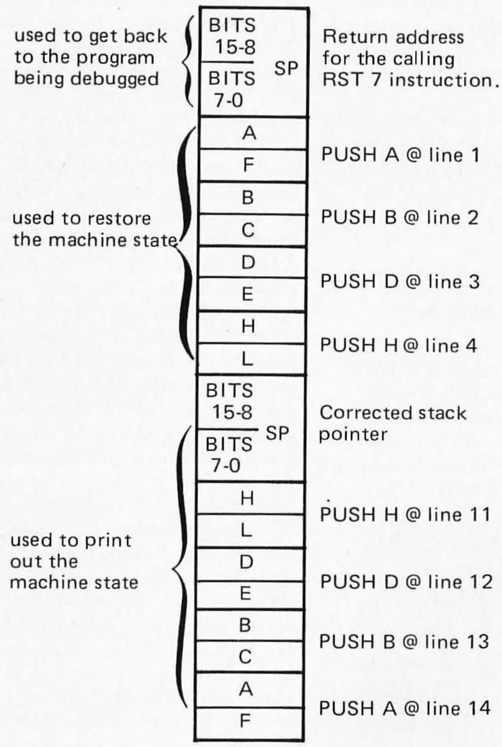


Figure 1: Stack Frame Used by TRAP. This diagram shows the allocation of data in the 8080's stack when TRAP routine is executing line 15 of listing 1. Actual contents vary during the course of TRAP, and this particular frame represents the maximum stack utilization (ignoring the supporting subroutine return address) during the routine. When TRAP routine returns, the state of the processor including the stack pointer is completely restored.

Continued from page 155

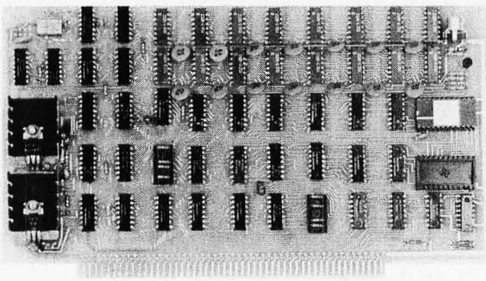
between inputs, data storage and subsequent retrieval, ASCII binary and BCD conversion of outputs, and a general purpose intelligent counter program allowing user interaction and function selection.

International Data Systems Inc is located at 400 N Washington St, Suite 200, Falls Church VA 22046. The kit form of the 88-UFC is available for $179.■

Circle 620 on inquiry card.

A High Resolution Text Display with Considerable Flexibility

Lorne Trottier, marketing manager for Matrox Electronic Systems, POB 56, Ahuntsic Station, Montreal, Quebec CANADA H3L 3N5, has sent us some information on a product which will be of great interest to users of many Altair,



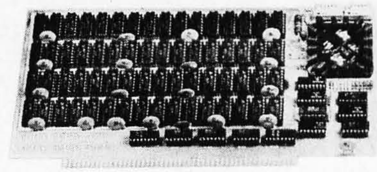
IMSAI, Poly 88 and other computers which have the Altair bus structure. This new product is the Model ALT-2480 text display, which produces a high resolution display of 80 character lines, 24 lines per screen. The character generator has several user selectable options including a reduced bandwidth mode with two pages of 40 character lines, 24 lines per page; 5 by 7 or 7 by 9 dot matrix character generators, upper or lower case letters, selectable reverse video or blinking, adjustable page boundary locations in address space, and either European or American standard television composite video outputs. By plugging it into an Altair or similar computer and running a coaxial cable to an appropriate video input, one can achieve a word processing output device with all the necessary features for text manipulation. In order to take advantage of the high resolution 80 by 24 character mode of operation, this display will require use of a relatively high bandwidth video monitor; it is not recommended for use with an RF converter and standard television set. This product is available in assembled form only, and is priced at $295, FOB Montreal CANADA. Delivery is quoted at two to four weeks after receipt of order.■

Circle 621 on inquiry card.

Two New Memory Boards

Advanced Microcomputer Products has announced two new memory cards which plug into an Altair (S-100) bus. These are the Logos I 8 K static low power memory board (photo 1) and the

Photo 1.



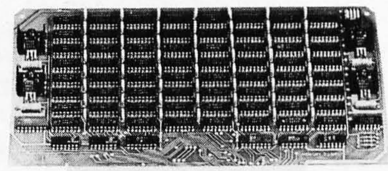


Photo 2.

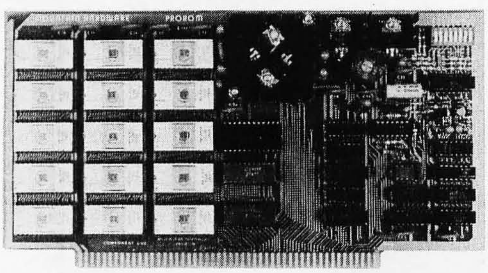
Model 801C 8 K memory board (photo 2). The Logos I board features switch selection of address space allocations at any 1 K boundary in memory address space, and has write protection options (1 8 K block, 2 4 K blocks, . . . 32 256

byte blocks), low 1.4 A power consumption and provision on the board for battery backup. Logos I is available as a kit for $248 ($31 per 1 K block installed) and $298 in assembled form ($37.25 per 1 K block installed). The Model 801C is a static memory board which can be placed at any 4 K boundary in memory address space, and trades a slightly lower price off against the lack of memory protect and battery backup features of the Logos I. The Model 801C is available in assembled form only, priced at $198 if 21L02-1 parts are used, and $207.95 if 91L02APC parts are specified for the memory array. Advanced Microcomputer Products is located at POB 17329, Irvine CA 92713.■

Circle 622 on inquiry card.

8 K Combined EROM and Volatile Memory Card

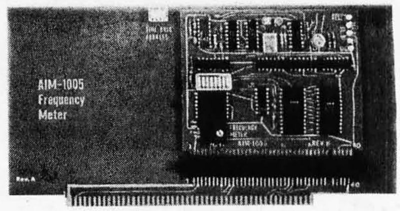
Mountain Hardware, Box 1133, Ben Lomond CA 95005, has introduced this PROROM board for use in computers which use the 100 pin Altair backplane design. The purpose of this board is to enable the user to store custom software in a region of 7.5 K bytes (7680 bytes decimal) of UV erasable read only memory parts. The board uses an even 8 K of memory address space, with the extra 0.5 K devoted to volatile programmable memory. The EROM parts employed in this design are AMI-S-6834 which can store 512 bytes per package in fifteen 24 pin packages. Also included in the design is a programming circuit which enables the user to write into the EROM regions. There are DIP switches for selecting the 8 K address range for the board, as well as write protect logic to protect against accidental writing into the EROM parts. The standard $164 kit price includes 256 bytes of volatile memory, one S-6834 EROM preprogrammed with an 8080 system monitor program, and complete documentation.



Where would this product prove most useful? Anyone considering assembly of a general purpose or dedicated application system with Altair compatible peripherals should find this card a useful component: its built-in volatile memory can provide the 8080's subroutine linkage stack, as well as a small amount of scratch pad storage for the application program; the 7.5 K bytes of read only memory capacity, when fully stuffed with EROM parts is sufficient for a quite sophisticated set of dedicated

Continued on page 168

A Frequency Meter Plug-In



The AIM-1005 is a novel Altair (S-100) bus interface which is an adaptation of a previously available industrial module to an Altair compatible 100 pin bus connector as seen in this picture. The AIM-1005 is primarily intended for use as a receiver of signals from transducers which generate frequency analogs of physical parameters for transmission to the computer over twisted pair wires. The precision of the measurement is 16 bits, and the company also provides the AIM-1000 transducer interface and a multiplexor for multichannel systems. This product was demonstrated to us on an Altair 8800a by Barry Hilton, its designer. He is also the proprietor of Automated Industrial Measurements Inc, POB 125, Wayland MA 01778.■

Circle 624 on inquiry card.

Circle 185 on inquiry card.

CRYSTALS

THESE FREQUENCIES ONLY

Part #	Frequency	Case/Style	Price
CY1A	1.000 MHz	HC33/U	$5.95
CY2A	2.000 MHz	HC33/U	$5.95
CY2.01	2.010 MHz	HC33/U	$1.95
CY3A	4.000 MHz	HC18/U	$4.95
CY7A	5.000 MHz	HC18/U	$4.95
CY12A	10.000 MHz	HC18/U	$4.95
CY14A	14.31818 MHz	HC18U	$4.95
CY19A	18.000 MHz	HC18/U	$4.95
CY22A	20.000 MHz	HC18/U	$4.95
CY30B	32.000 MHz	HC18/U	$4.95

XR-2206KB Kit $29.95 Special XR-2206KA Kit $19.95

EXAR

WAVEFORM GENERATORS		MISCELLANEOUS		TIMERS	
XR-205	$8.40	XR-2211CP	$6.70	XR-555CP	$ .39
XR-2206CP	4.49	XR-4136	.99	XR-320P	1.55
XR-2207CP	3.85	XR-1468	3.85	XR-556CP	1.85
STEREO DECODERS		XR-1488	5.80	XR-2556CP	3.20
XR-1310CP	$3.20	XR-1489	4.80	XR-2240CP	3.25
XR-1310EP	3.20	XR-2208	5.20	**PHASE LOCKED LOOPS**	
XR-1800P	3.20			XR-210	5.20
XR-2567	2.99			XR-215	6.60
				XR-567CP	1.95
				XR-567CT	1.70

CONNECTORS

PRINTED CIRCUIT EDGE-CARD

.156 Spacing-Tin-Double Read-Out
Bifurcated Contacts — Fits .054 to .070 P.C. Cards

15/30	PINS (Solder Eyelet)	$1.95
18/36	PINS (Solder Eyelet)	$2.49
22/44	PINS (Solder Eyelet)	$2.95
50/100 (.100 Spacing)	PINS (Solder Eyelet)	$6.95

25 PIN-D SUBMINIATURE

DB25P	PLUG	$3.25
DB25S	SOCKET	$4.95

IMC 3½ DIGIT DVM KIT



This 0-2 VDC .05 per cent digital voltmeter features the Motorola 3½ digit DVM chip set. It has a .4" LED display and operates from a single +5V power supply. The unit is provided complete with an injection molded black plastic case complete with Bezel. An optional power supply is available which fits into the same case as the 0-2V DVM allowing 117 VAC operation.

A. 0-2V DVM with Case $49.95
B. 5V Power Supply $14.95

Vector

Etching Kits

- 32 X A-1 — P.C. Etch Materials Kit enough for 5 circuit boards — $29.95 ea.
- 27 X A-1 — Etched Circuit Kit Complete kit — only add water — $ 9.95 ea.

Plugboards

- 3662 — 6.5 X 4.5 X 1/16 Epoxy glass P-Pattern-44 P.C. Tabs-spaced .156" — $ 6.95 ea.
- 8800V — Universal Microcomputer/Processor plugboard — Epoxy Glass — complete with heatsink and mounting hardware 5.313 X 10 X 1/16 copper clad — $19.95 ea.

1/16 VECTOR BOARD



	0.1" Hole Spacing Part No.	P-Pattern L	P-Pattern W	Price 1	Price 2-Up
PHENOLIC	64P44 062XXXP	4.50	6.50	1.72	1.54
	169P44 02XXXP	4.50	17.00	3.69	3.32
EPOXY GLASS	64P44 062	4.50	6.50	2.07	1.86
	84P44 062	4.50	8.50	2.56	2.31
	169P44 062	4.50	17.00	5.04	4.53
	169P84 062	8.50	17.00	9.23	8.26
EPOXY GLASS COPPER CLAD	169P44 062C1	4.50	17.00	6.80	6.12

HEAT SINKS



205CB 680-.75A



205-CB	Beryllium Copper Heat Sink with Black Finish for TO-5	$ .25
291-.36H	Aluminum Heat Sink for TO-220 Transistors & Regulators	$ .25
680-.75A	Black Anodized Aluminum For TO-3	$1.60

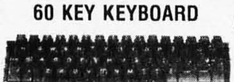
HEXADECIMAL ENCODER 19-KEY PAD



- 1 - 0
- ABCDEF
- Return Key
- Optional Key (Period)
- — Key

$10.95 each

60 KEY KEYBOARD



This keyboard features 60 unencoded SPST keys, unattached to any kind of P.C.B. A very solid molded plastic 13" x 4" base suits most applications.

$19.95

HD0165 16 LINE TO FOUR BIT PARALLEL KEYBOARD ENCODER $7.95

TOOLS

A97MS	— Diagonal Cutter - 4" semi-flush cut	$8.50 ea.
A11DMS	— Chain Nose Pliers - 4¾" long	7.50 ea.
T-6	— Wire Stripper - #16 to #26 gauge	3.75 ea.
55B	— Wire Stripper - #10 to #20 gauge	2.50 ea.
CS-8	— Cutter-Crimper Tool - 8¼" long	8.50 ea.
Nippling Tool	— Cuts, Trims or Notches Metal up to #18 gauge	6.95 ea.
Nibbling Tool Replacement Punch		3.75 ea.

PERMACEL® P-29 PLUS Electrical Tape - All Weather
• ¾" wide x 66 ft. • Black vinyl
1-9 Rolls **$.79 each** 10-up Rolls **$7.95/10 roll package**

MICROPROCESSOR COMPONENTS

8080A	CPU	$19.95	8228	System Controller - Bus Driver	$10.95
8212	8 Bit Input/Output	4.95	MC6800L	8 Bit MPU	35.00
8214	Priority Interrupt Control	15.95	MC6820L	Periph. Interface Adapter	15.00
8216	Bi-Directional Bus Driver	6.95	MC6810AP1	128 x 8 Static RAM	6.00
8224	Clock Generator/Driver	10.95	MC6830L8	1024 x 8 Bit ROM	18.00
CDP1802	- with user manual	39.95	Z80	CPU	49.95

CPU'S

8080	Super 8008	24.95
8080A	Super 8008	19.95
2650	8 BIT MPU	26.50

SR'S

2504	1024 Dynamic	$ 3.95
2518	Hex 32 BIT	7.00
2519	Hex 40 BIT	4.00
2524	512 Dynamic	2.49
2525	1024 Dynamic	6.00
2527	Dual 256 BIT	3.95
2529	Dual 512 BIT	4.00
2532	Quad 80 BIT	3.95
2533	1024 Static	Special 5.95
3341	Fifo	6.95
74LS670	16 x 4 Reg	3.95

UART'S

AY-5-1013	30K Baud	$5.95

ROM'S

2513(2140)	Char. Gen.-upper case	$ 9.95
2513(3021)	Char. Gen.-lower case	9.95
2516	Char. Gen.	10.95

RAM'S

1101	256 x 1	Static	$ 1.49
2101	256 x 4	Static	5.95
2102	1024 x 1	Static	1.75
2107/5280	4096 x 1	Dynamic	4.95
2111	256 x 4	Static	6.95
7489	16 x 4	Static	2.49
8101	256 x 4	Static	6.95
8111	256 x 4	Static	6.95
8599	16 x 4	Static	3.49
91L02	1024 x 1	Static	2.25
74200	256 x 1	Static	6.95
93421	256 x 1	Static	2.95
MM5262	2K x 1	Dynamic	2 for 1.00

PROMS

1702A	2048	Famos	$ 9.95
5203	2048	Famos	14.95
82S23	32 x 8	Open C	5.00
82S123	32 x 8	Tristate	5.00
74S287	1024	Static	7.95
3601	256 x 4	Fast	3.95
2708	8K	Eprom	29.95
6301-1	1024	Tri-State Bipolar	3.49
6330-1	256	Open Collector Bipolar	2.95
6331-1	256	Tri-State Bipolar	2.95

SPECIAL REQUESTED ITEMS

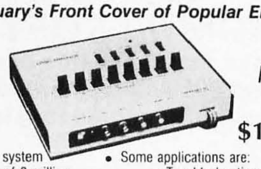
FCM3817	$ 5.00						
AY-3-8500-1	$16.95					8T97	2.00
MC3061P	3.50	CD4508	6.75	82S115	25.00	3341	6.95
MC4016P (74416)	7.50	CD4515	6.50	5841	9.95	9368	3.95
MC14583	3.50	CD4520	2.70	MK50240	17.50	MC1408L7	9.95
MC14562	14.50	MCM6571	17.50	11C90	19.95	LD110/LD111	25.00/set
CD4059	9.95	MCM6574	17.50	DS0026CH	3.75	AY-5-9100	17.50 ea.
CD4070	.95	MCM6575	17.50	TIL308	10.50	95H90	13.95
MC14409	14.95	MC14419	14.95	ICM7208	22.00	ICM7209	7.50
MC14410	14.95	ICM7045	24.95	ICM7207	7.50	HD0165	7.95

PARATRONICS

ALLOW 1 TO 3 WEEKS DELIVERY

Featured on February's Front Cover of Popular Electronics

Logic Analyzer Kit



MODEL 100A
$189.00/Kit

- Analyzes any type of digital system
- Checks data rates in excess of 8 million words per second
- Trouble shoot TTL, CMOS, DTL, RTL, Schottky and MOS families
- Displays 16 logic states up to 8 digits wide
- See ones and zeros displayed on your CRT, octal or hexadecimal format
- Tests circuits under actual operating conditions
- Easy to assemble — comes with step-by-step construction manual which includes 80 pages on logic analyzer operation.

- Some applications are:
  - — Troubleshooting microprocessor address, instruction, and data flow
  - — Examine contents of ROMS
  - — Tracing operation of control logic
  - — Checking counter and shift register operation
  - — Monitoring I/O sequences
  - — Verifying proper system operations during testing

BUGBOOK ®

Continuing Education Series



BUGBOOK I & II - Basic concepts of TTL Logic — over 90 experiments **$17.00/set**

BUGBOOK IIa - Introduces UART — recommended for RTTY enthusiast **$5.00/book**

BUGBOOK III - Explores 8080 chip — introduces Mark 80 Microcomputer **$15.00/book**

555 TIMER APPLICATIONS SOURCEBOOK WITH EXPERIMENTS — over 100 design techniques **$6.95/book**

CMOS-M-DESIGNERS PRIMER AND HANDBOOK
a complete CMOS instruction manual **$6.00**

Introductory Offer — all 6 books (worth $49.95)
SPECIAL - $42.95

CONTINENTAL SPECIALTIES

PROTO BOARD 6
$15.95
(6" long X 4" wide)



Other CS Proto Boards

PB100 - 4.5" x 6"	$ 19.95
PB101 - 5.8" x 4.5"	29.95
PB102 - 7" x 4.5"	39.95
PB103 - 9" x 6"	59.95
PB104 - 9.5" x 8"	79.95
PB203 - 9.75 x 6½ x 2¾	75.00
PB203A - 9.75 x 6½ x 2¾ (includes power supply)	120.00

Logic Monitor $84.95
for DTL, HTL, TTL or CMOS Devices

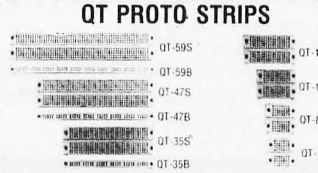
PROTO CLIPS

14 PIN	$4.50
16 PIN	4.75
24 PIN	8.50

DESIGN MATES

- DM1 - Circuit Designer 54.95
- DM2 - Function Generator 69.95
- DM3 - RC Bridge 59.95

QT PROTO STRIPS



QT type	#holes	price
QT-59S	590	12.50
QT-59B	bus strip	2.50
QT-47S	470	10.00
QT-47B	bus strip	2.25
QT-35S	350	8.50
QT-35B	bus strip	2.00
QT-18S	180	4.75
QT-12S	120	3.75
QT-8S	80	3.25
QT-7S	70	3.00

Experimentor 300	$ 9.95
Experimentor 600	$10.95

$5.00 Minimum Order — U.S. Funds Only
California Residents — Add 6% Sales Tax

Spec Sheets - 25¢ — Send 35¢ Stamp for 1977A Catalog
Dealer Discount Available — Request Pricing

NOW OPEN SATURDAYS

James ELECTRONICS

1021-A HOWARD AVE., SAN CARLOS, CA. 94070
PHONE ORDERS WELCOME — (415) 592-8097
All Advertised Prices Good Thru August

Timeband

A Trademark of Fairchild Camera and Instrument Corporation

Digital Alarm Clocks



- 24-hour alarm
- Doze Button
- 100% Solid State
- Large Red Led Display (.8" high)
- AM/PM Indicator
- Seconds display at touch of button
- Power failure indicator
- One year factory warranty

C-500 - Ivory Case
C-500B - Ebony Case
$16.95



C-8211
Woodgrain Case
$19.95

DIGITAL AUTO INSTRUMENT

SEVEN DIFFERENT INSTRUMENTS!
MEETS OR EXCEEDS ORIGINAL AUTOMOTIVE SPECS.

*Please specify which **one** of the seven models you want when ordering – these **do not** all come in one unit. Each model must be bought separately.*



1. TACHOMETER 0-9900 RPM 4, 6 or 8 Cyclinders
2. WATER TEMP. 100-250° F
3. FUEL LEVEL Percentage Low Fuel Indicator
4. SPEEDOMETER* 0-99 MPH
5. OIL PRESSURE 0-80 PSI
6. OIL TEMP 200-350° F
7. BATTERY MONITOR ±1 Volts From 11-15 VDC

BRIGHT YELLOW ORANGE 3" LED DISPLAY!

Kit includes case, bracket and all components — complete. Nothing else to buy! 12 Volt NEG GRD.
DIMENSIONS: 4½ x 4 x 2 **KIT: $49.95**
* Add $10.00 for required speed transducer. **ASSEMBLED: $59.95**

DIGITAL STOPWATCH

- Bright 6 Digit LED Display
- Times to 59 minutes 59.59 seconds
- Crystal Controlled Time Base
- Three Stopwatches in One
  Times Single Event — Split & Taylor
- Size 4.5" x 2.15" x .90" (4½ ounces)
- Uses 3 Penlite Cells.



Kit — $39.95
Assembled — $49.95
Heavy Duty Carry Case $5.95

Stop Watch Chip Only (7205) $19.95



AUTOTEL — an audible alarm kit indicating potential engine damage. An audible signal (70 db pulsing) immediately forewarns a malfunction or failure. There is no sound during normal operation. Features CMOS circuitry. 2½" x 2½" x ⅞" case. Complete kit with all components, hardware and case.
SPECIAL $6.95/Kit

QUARTZ DIGITAL AUTO CLOCK OR ELAPSED TIMER!

Elapsed Timer: Hrs, Mins and Secs
12 or 24 Hr Capacity
Simple Reset - Start Pushbutton Control

Complete kit includes mounting bracket, case and all components, nothing else to buy. Features MM5314 chip: Large .4" LED's. Accuracy better than ± min. per mo. internal battery backup. 12 volt non-polar operation.



DIMENSIONS 4½" x 4" x 2"
12 or 24 HOUR MODE

Kit: $29.95
Assembled: $39.95

CASE ONLY (includes hardware, mounting bracket and bezel) **$6.50**

JE700 CLOCK



12 or 24 Hour
115 VAC

The JE700 is a low cost digital clock, but is a very high quality unit. The unit features a simulated walnut case with dimensions of 6" x 2½" x 1". It utilizes a MAN72 high brightness readout, and the MM5314 clock chip.

$17.95



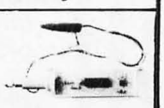
DIGITAL CLOCK KIT — 3½ INCH DIGITS

4 DIGIT KIT $49.95 **4 DIGIT ASSEMBLED $59.95**
6 DIGIT KIT $69.95 **6 DIGIT ASSEMBLED $79.95**

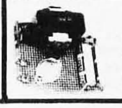
This clock features big 3½" high digits for viewing in offices, auditoriums, etc. Each digit is formed by 31 bright 0.2" LED's. The clock operates from 117 VAC, has either 12 or 24 hr. operation. The 6 digit version is 27" x 3½" x 1½" and the 4 digit is 18" x 3½" x 1½". Kits come complete with all components, case and transformer.
Specify 12 or 24 Hour When Ordering

JE803 PROBE

The Logic Probe is a unit which is for the most part indespensible in trouble shooting logic families TTL, DTL, RTL, CMOS. It derives the power it needs to operate directly off of the circuit under test, drawing a scant 10 mA max. It uses a MAN3 readout to indicate any of the following states by these symbols: (H) - 1 (LOW) - 0 (PULSE) - P. The Probe can detect high frequency pulses to 45 MHz. It can't be used at MOS levels or circuit damage will result.



$9.95 Per Kit
printed circuit board



T²L 5V 1A Supply

This is a standard TTL power supply using the well known LM309K regulator IC to provide a solid 1 AMP of current at 5 volts. We try to make things easy for you by providing everything you need in one package, including the hardware for only **$9.95 Per Kit**

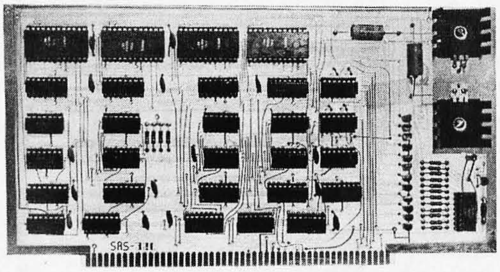
Circle 187 on inquiry card.

Continued from page 164

applications or systems programs. If more scratch memory were needed, any one of a number of Altair compatible 4 K, 8 K or 16 K memory cards could be used with the resident nonvolatile EROM capacity of this card.■

Circle 625 on inquiry card.

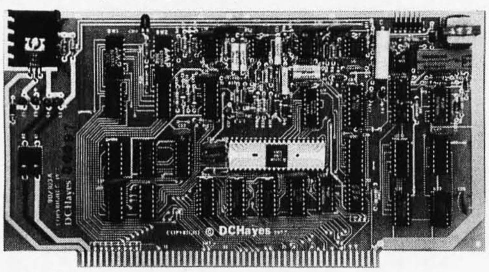
A Noteworthy New Board



Mary P Rezanow, production manager of Stillman Research Systems, POB 14036, Phoenix AZ 85063, sent along information on the new SRS-320 Generator Board. This music peripheral is used to create four distinct channels of output for an Altair compatible computer, with each channel consisting of a well tempered musical pitch timing signal set by a 4 bit note code and a 4 bit octave code. In the minimal music system represented by this board, the outputs of the four channels are summed as square waves and can be used to play four part polyphonic music [*with a timbre most appropriate for baroque organ music*]. The board comes with a complete manual, including two 8080 programs and hexadecimal music texts for "Joy" by J S Bach (one part only) and "Dona Nobis Pacem," composer unknown (polyphonic). The two sample texts and the two different driver programs are intended for use on minimal systems. The company is designing a second module in this series, which will be called a Modifier Board and will take the pitch outputs of the Generator Board and process them in a manner similar to a conventional music synthesizer but with computer control of parameters.■

Circle 626 on inquiry card.

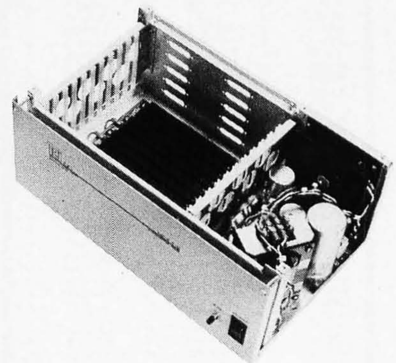
Do You Have Some Data to Communicate?



D C Hayes, located at POB 9884, Altanta GA 30319, has introduced this Model 80-103A Data Communications Adapter for the Altair (S-100) bus. The product is fully programmable and implements automated dialing or answering, 110 and 300 bits per second data rates with character format and parity error detection, uses the standard US telecommunications modem frequencies, and is fully compatible with the Bell System when attached via a DAA device. The unit features crystal control and all digital modulation and demodulation, so there are no adjustments to be made. Users seriously interested in networking of personal computers will want to consider this product, which is available in several forms: fully assembled and tested at a retail price of $279.95, as a bare board (sans electronic components) with manual for $49.95, and as the manual alone for $7.50. Dealer inquiries are invited as well.■

Circle 630 on inquiry card.

Attention Altair Bus Experimeters...



Here is an interesting new product which allows the experimenter, or systems consultant, to completely start from scratch in the selection of Altair (S-100) bus compatible peripherals and system components. This new product, called The Mainframe, is the physical starting point for such a system with its essential physical and electrical characteristics. The Mainframe includes a heavy duty vented aluminum cabinet finished in blue color, a 12 slot Altair compatible mother board (with 100 pin connectors inserted, soldered and checked out), a front panel which includes an indicating AC switch and a reset switch, a unique power supply which includes a constant voltage transformer for brownout protection, better than 100 db isolation of the DC supply voltages from AC line transients, and a capacity of 17 A at 8 V and 2 A at +16 and -16 V. A muffin fan rated at 115 cubic feet (3.26 cubic meters) per minute air flow is mounted in the cabinet. All wiring of the kit version is precut and prelugged for ease of assembly.

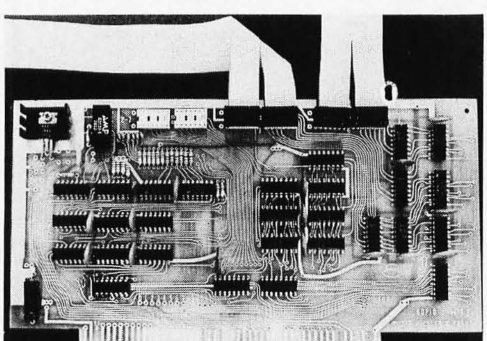
No price was given in the press release information we received, so potential users should contact CMC Marketing Corporation at 7231 Fondren Rd, Houston TX 77036. Specify either Model MCS-112K or MCS-112A for the kit or assembled versions, respectively.

This product will prove useful to people who want to select a processor board from manufacturer X, a systems software read only memory and terminal interface board from manufacturer Y, main volatile memory boards from manufacturer Z, and whiz-bang peripheral board from manufacturer Q. The result can be a completely custom selection of boards.■

Circle 628 on inquiry card.

And Now, an IEEE-488 Standard Bus Interface

Pickles & Trout have announced this new peripheral interface for the Altair (S-100) bus which allows any one of the many computers which have copied the Altair design to talk to instruments and peripherals which use the Hewlett-Packard Interface Bus, now supported by the Institute of Electrical and Electronic Engineers (IEEE) as IEEE standard 488. While the 488 standard was originally intended for laboratory instrumentation, its generality makes it quite useful in personal computer systems. A number of commercial and industrial peripherals are already on the market which use this communications discipline.



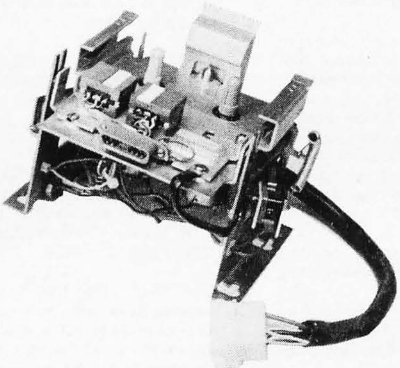
Six of the eight ports on the Pickles & Trout BDPIO board (Bi-Directional Parallel Input and Output) are on the bi-directional IEEE 488 bus which includes six interrupt lines and an uncommitted line for power or other user selected functions. This bus allows a small number of lines (25) to service devices which require several ports, such as the Digital Group cassette controller. The six ports require only one cable assembly and use a single 25 pin D style connector. An external junction box also available from P & T is used to decode addresses on the 488 bus at the peripheral, and provide additional signal buffering for systems with long cables. For more information on this interesting peripherals interface line, contact Pickles & Trout at POB 2270, Goleta CA 93018.■

Circle 629 on inquiry card.



FIRE-SMOKE-INTRUDER ALARM

12 volt DC operation, large 5 inch bell, one unit, all the works under the bell. Nice for camper, boat, homes. $25



VIATRON CASSETTE DECKS

The computer cassette deck alone $35 Set of 2 boards read/write amp & serve control boards for this deck $40.00



PORTABLE FIRE ALARM

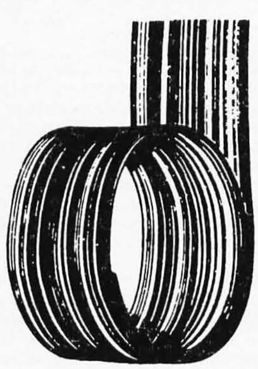
Operates on 3 internal AA cells (not furnished) On temp rise of approx 150 Sonalert type alarm sounds off. Push to test switch makes nice sounding code practice oscillator $6.00

SPECTRA FLAT TWIST

50 conductor, 28 gauge, 7 strands/conductor made by Spectra. Two conductors are paired & twisted and the flat ribbon made up of 25 pairs to give total of 50 conductor. May be peeled off in pairs if desired. Made twisted to cut down on "cross talk." Ideal for sandwiching PC boards allowing flexibility and working on both sides of the boards. Cost originally $13.00/ft

SP-324-A $1.00/ft. 10 ft/$9.00

SP-234-A $1.00 ft 50 cond. 10 ft/$9.00
SP-234-B .90 ft 32 cond. 10 ft/$8.00



WIRE WRAP WIRE

TEFZEL blue #30 Reg. price $13.28/100 ft. Our price 100 ft $2.00; 500 ft $7.50.

MULTI COLORED SPECTRA WIRE

Footage			10′	50′	100′
8 Cond.		#24	$2.50	9.00	15.00
12	"	22	3.00	11.00	18.00
14	"	22	3.50	13.00	21.00
24	"	#24	5.00	20.00	30.00
29	"	22	7.50	28.00	45.00

Great savings as these are about 1/4 book prices. All fresh & new.

TOUCHTONE ENCODER CHIP

Compatible with Bell system, no crystal required. Ideal for repeaters & w/specs. $6.00

CHARACTER GENERATOR CHIP

Memory is 512x5 produces 64 five by seven ASCII characters. New material w/data $6.00

Meshna

Please add shipping cost on above. Minimum order $10
FREE CATALOG SP-9 NOW READY
P.O. Box 62, E. Lynn, Massachusetts 01904

Circle 188 on inquiry card.

Classified Ads

FOR SALE: One Central Data 2650 board with 8 K of static programmable memory, power supply and cabinet. Has connectors for the keyboard, video monitor and tape recorder, and is totally assembled and tested; best offer over $850. Also have a switch console for the 2650 board. Has address, data and control bus switches as well as hardware breakpoint and single step. $250. J Roloff, 2214 Brookshire Dr, Champaign IL 61820.

USER'S GROUP: For 2650-based microprocessor as featured in April 1977 *Radio Electronics*. G Frank Humiston, 506 Wayne Av, El Cajon CA 92021.

FOR SALE: Hewlett-Packard HP-55 scientific programmable calculator with all original accessories, plus a security cradle, extra recharger, and reserve power pack (less battery). Asking $225. Frank Scavo, 1811 Court St, Syracuse NY 13208.

FOR SALE: Five new Phi-Deck units. $100 each, $25 with order, balance COD. Ron Riley, POB 4310, Flint MI 48504.

FOR SALE: One IMSAI 8080 microcomputer with 6 slot mother board, 4 K RAM, and IMS MIO multiple IO board with cassette interface. Also one cable for one IO port. All assembled and working. Postpaid $965. Michael C Bechard, 1 Walnut St, Box 242, Champlain NY 12919.

FOR SALE: Burroughs Accounting System. Consists of E6090 processor, E6000 console, A4004 ledger reader, A594 card reader, A988 line printer. Magnet wire #29 to #35, $2 lb; #36 to #40, $2.50 lb; #41 to #44, $3 lb, SASE for reply. Douglas Craton, 5625 Balfrey Dr, W Palm Beach FL 33406, (305) 686-1106 after 6:00 PM.

WANTED: Looking for PDP-8 m/e modules, positive IO, Teletype, Data Break, 8 K memory, and RK 8 controller. Who can help? Wolf L Carstens, D-2359 Hasenmoor, W GERMANY.

WANTED: Altair 8800, 8800a, or 8800b, new or used, kit or assembled, in working order, at a good price. I also need one copy each of BYTE issues 1 thru 11. Call evenings (201) 652-2875 or write D Rubin, 47 Dale Av, Wyckoff NJ 07481.

FOR SALE: HP-65 fully programmable calculator with all accessories, like new, with two Math Pacs, Nav Pac 1 and Stat Pac 1, Standard Pac, blank magnetic cards, complete HP-65 User's Library catalog, $325. Jack Sullivan, 17 Pelham Rd Hudson NH 03051, (603) 883-8514 evenings.

FOR SALE: SWTPC CT-1024 converted to 64 characters, computer cursor, screen read, power supply, working perfectly, best offer over $250. MARK-8: output port, CPU board, address latch, display board, input multiplex, 1 K memory without chips, C8008-1, documentation, populated boards, best offer over $120. Will wait two weeks, return all others. I pay postage. Cashiers check, money order, postal money order only. Matt Perdue, 4015 N 20th Rd, Arlington VA 22207, (703) 527-8979.

Readers who have equipment, software or other items to buy, sell or swap should send in a clearly typed notice to that effect. To be considered for publication, an advertisement should be clearly noncommercial, typed double spaced on plain white paper, and include complete name and address information. These notices are free of charge and will be printed one time only on a space available basis. Insertions should be limited to 100 words or less. Notices can be accepted from individuals or bona fide computer users clubs only. We can engage in no correspondence on these and your confirmation of placement is appearance in an issue of BYTE.

Please note that it may take three or four months for an ad to appear in the magazine. ■

FOR SALE: Two complete CDC, RPC-4000 computers, each includes CDC paper tape reader, Friden paper tape punch, Royal McBee console typewriter 8000, 32 bit drum memory. These are second generation computers, with discrete transistors, etc. Both are running. With the above goes: one Friden adding machine with paper tape punch; one Friden Flexowriter, an extra Friden punch, a Soroban high speed punch, and several other peripherals partially scavenged for parts. Each computer was $80,000 when new. Will sell the works for $2,500 including a compiler, some utility programs and some application programs. Walter E Page, 3960 S Marginal Rd, Cleveland OH 44114.

FOR SALE: Viatron System 21, complete with CPU power supply, keyboard, CRT monitor, and two tape drives, $475. Also Friden 10 cps paper tape reader-punch with 80 column card reader attached, $95. Michael Toft, 69 Westbury St, Thousand Oaks CA 91360, (805) 497-7141.

WANTED: DECtape read write head in usable condition preferably with cable and Amphenol connector intact. Also CM8/I card reader with manuals and engineering drawing if possible. Quote price on any or all as well as other PDP-8/I peripherals, memory, parts you may wish to put into our kindly "old horse" pasture. FREE: copy of Mark-8 Construction Booklet for 10 by 12 inch. SASE. Bro Felix Neussendorfer, Colegio San Antonio Abad, POB 729, Humacao PR 00661.

FOR SALE: Typagraph Terminal (a modified Model 33 Teletype). Good condition. Full plotting capability. Will make beautiful curves. Can make 1/50 inch incremental moves in any direction. Also has paper tape punch and reader. Two electronic modules included. Software available. $995. Call Nancy Daedone (408) 279-2737.

FOR SALE: Brand new MITS 88-S4K synchronous dynamic memories, professionally built by electronic company, $130 each. Call days. Marco Lanza (617) 244-3216.

WANTED: Schematic or technical assistance on interfacing a Friden Flexowriter (Model SPS) to an IMSAI 8080. Would like to be able to use the Friden in both local or remote mode. The IMSAI has both serial and parallel ports. Write John Gill, RT 5 Box 370, Blountville TN 37617.

FOR SALE: Portable TV, modified for TVT use $55; MITS 1 K static assembled, best offer; 4 K static $95; MITS 4 K static $125; HP line printer 11 columns, digits only, needs interfacing $40 Charles Zalenski, POB 92, Johnson City NY 13790.

FOR SALE OR TRADE: One SIO IMSAI board, two ports. New, never used, assembled as a kit tested by IMSAI. One Carterfone data modem, half full and power, RS-232; some parts need replacement but in good condition. Make an offer. Garry Baratta, POB 335, Eveleth MN 55734, or call (218) 741-6838 after 7:00 CST.

FOR SALE: Altair 680b, assembled and running great. Best offer over $300. Alan Dahlbom, 16 Forest St, Cambridge MA, (617) 492-0676, evenings.

FOR SALE: Friden Model TM20K714 keyboard-printer with TM20K15 controller and schematics. $300, you pay shipping. Also copies of schematics for above controller available for $6 to cover copying and first class shipping. Tim Ahrens, 2200 Sorret Tree Ct, Austin TX 78744.

FOR SALE: IMSAI 8080, 8 K IMSAI memory, 22 slot mother board, IMSAI audio cassette board, Processor Tech 3P+S interface, Oliver Audio Engineering OP-80A paper tape reader, plus much miscellaneous software, plugs, sockets, etc. All in like new condition, $899. Ken Schweim, 1709 Mary Ln, N Mankato MN 56001. (507) 388-5734.

FOR SALE: Percom data cassette interface CI-810 in box with cable and connector, ready to plug into a parallel IO port, $50. Larry Belmontes Jr, 1762 Yale St, Corpus Christi TX 78416.

FOR SALE: OP-80 paper tape reader, brand new and working, used only three times; regularly $80, asking $55. Larry Belmontes Jr, 1762 Yale St, Corpus Christi TX 78416.

WANTED: First ten issues of BYTE. K C Naizer, POB 355, Nederland TX 77627.

TELETYPE: Western Union Model 101 Baudot Teleprinter. Going to college, no room for machine, good condition. Sell for $60 or trade for ??? David J Lilja, 6401 Van Wert Rd Rt 3, Cedar Falls IA 50613, (319) 266-7633.

SWAP OR SELL: Have first 16 issues of BYTE, like new. Desire Grid Dip Meter, ham radio equipment, or ??? Billy Travis, 4054 Sawtelle Blvd, Los Angeles CA 90066.

WANTED: Micropersonal computing mate. I've gotten so wrapped up with microprocessors that my love life is suffering. If there's an interesting female out there with similar interests, let's get synergistic. I fly, so a reasonable distance is no problem. Martin Edwards, POB 6244, Torrance CA 90504 or answering service telephone (213) 874-2200.

WANTED: Burned out ICs, any size. Also burned out CPUs in 40 dip package. Indicate price desired. Is there a computer club for 1802 users? If so, tell me how I can join. Jeff Foster, POB 603, Bellflower CA 90706.

FOR SALE: ASR 33 TTY with auto reader, auto punch, print blind features, parts, maintenance, lube manuals and diagrams, good condition $700. A W Walker, 6000 N 27th St, Arlington VA 22207, (703) 533-9336.

FOR SALE: IBM 024 Card Punch, with alphanumeric keyboard, good condition with documentation $275. IBM 056 Card Verifier, with alphanumeric keyboard, documentation, good condition $125. Use for original intended purpose or modify readily for microprocessor card IO. Will not ship. A W Walker, 6000 N 27th St, Arlington VA 22207, (703) 533-9336.

FOR SALE: 63 key ASCII encoded keyboard, 100% working and built, $50. Oliver Audio Engineering OP-80A paper tape reader assembled and working, $75. Memory Translator, digital FSK modem (see April 1975 *Popular Electronics*) $75. Call (617) 963-5578 Monday thru Friday after 3PM.

WANTED: Issues 1 thru 16 of BYTE in good condition. Best offer under $75. Brian Branson, POB 911, Loma Linda CA 92354, (714) 796-1128.

FOR SALE OR SWAP: Texas Instruments SR-52 with PC-100. Outfit includes all standard accessories, EE1 program library, PPX-52 membership, and spare battery pack. As a complete outfit only, will sell for $340 or swap with cash for HP-67. David Lamkins, 502-12 Sherman St, Canton MA 02021, (617) 828-7480 or 253-2460 most days and evenings.

FOR SALE: Complete Viatron System 21 with communications adaptor (missing UART); includes all data and about six Viatapes, $375. Printing robot and interface to Viatron, $175. IBM Selectric Model 72 typewriter, $300. Will take $730 for all or make offer. All in perfect operating condition; you ship. Keith Teague WB5FMJ, POB 65, Alma AR 72921, (501) 632-2532 after 6 PM.

ATTENTION HAMS: I have vintage *QST* and *CQ* magazines, 1962 to 1966, and I will swap ten for one BYTE issues #1, 2, 3, 8, 9, 14. Dan Lasley, 3365 Steve #14, Memphis TN 38111, (901) 744-4866.

FOR SALE: Issues 2, 3, 4, 5 of BYTE; two MITS 4 K programmable memory boards assembled and in excellent condition; one 5 V, 25 A Wanless regulated power supply; one Heath IMPScope. Will accept best offer on each item. David Milhouse, 2823 Griffa Av, Columbus IN 47201.

SELL OR TRADE: PDP-8 4 K minicomputer with TTY interface $750. Will sell with Teletype for $1500. Data Mark 300 1 pm printer with PDP-8 interface and spare memory module $900. PDP-8/E Teletype interface $200. Ampex TM-11-291 9 track 150 ips tape drive $600. Wayne Simpson, 3330 W 64th Pl, Tulsa OK 74132, (918) 446-0401.

Circle 189 on inquiry card.

World's Lowest IC Prices

* SPECIAL PRICES *

MEMORIES

Rams

74S200	2.95
2102	1.40 *
2102-1	1.60 *

Proms

82S23/S123	1.95 *
82S129	3.25

Others

TMS 3409	2.00
MM 5013	1.50
NS 5260	1.50
NS 8619	2.00
MH 0026H	3.25

TTL

7400	.12 *
7402	.14
7403	.14 *
7404	.16 *
7407	.20
7410	.12 *
7416	.25
7420	.12 *
7427	.25
7437	.20
7438	.20 *
7440	.12 *
7441	.65 *
7445	.60

7447	.65 *
7450	.14
7451	.14 *
7473	.28
7474	.28
7475	.40
7480	.35 *
7483	.68
7486	.28
7490	.45
7493	.44 *
7495	.49
74107	.29
74109	.30
74116	1.50
74123	.45 *
74141	.80 *
74145	.65 *
74150	.60 *
74151	.60
74152	.90
74155	.60
74157	.60
74160	.75
74161	.60 *
74163	.75 *
74165	.80

74173	1.25
74174	.75
74175	.75 *
74177	.70
74180	.65 *
74181	1.50
74191	.85
74192	.70 *
74193	.70 *
74194	.85
74195	.68
74198	1.25
9602	.50 *
9300	.75
9312	.70

SCHOTTKY

74S01	.25
74S02	.25
74S37	.40
74S38	.60
74S85	2.00
74S113	.80
74S138	1.50
74S139	1.50
74S140	.50
74S151	2.00
74S153	2.50
74S172	4.50
74S174	2.05
74S175	2.05 *
74S181	2.95
74S197	2.20
74S257	1.50

LINEARS

DM8820/30	1.75
NE536T	2.75
NE555V	.43
NE556A	.90
1456V	.75
1458V	.52
566V	1.25
567V	1.35
540L	2.00

HIGH SPEED

74H00	.20
74H01	.20
74H04	.20
74H10	.20
74H11	.20
74H40	.20
74H51	.20
74H52	.20
74H74	.40
74H103	.50
74H106	.50

CMOS

4001	.16
4002	.16
4006	.90
4007	.16
4008	.70
4011	.16 *
4012	.16 *
4013	.30 *
4015	.80
4016	.35 *
4019	.70
4020	.90
4021	.95
4023	.16 *
4025	.20 *
4027	.40
4028	.60
4030	.35
4040	.95
4042	.60
4043	.75
4044	.70
4049	.35 *
4050	.35 *
4066	.65
4068	.35
4071	.16
4073	.16
4075	.16
4516	.85
4528	.75

LOW POWER SCHOTTKY

74LS00	.23 *
74LS02	.23 *
74LS08	.23 *
74LS10	.23 *
74LS27	.25 *
74LS73	.45
74LS75	.65
74LS151	.75 *
74LS153	.75 *
74LS157	.75 *
74LS161	1.00 *
74LS163	1.50
74LS164	1.50
74LS174	1.00 *
74LS175	1.50
74LS193	1.50
74LS221	1.25 *
74LS251	1.50
74LS253	1.50
74LS257	1.25 *
74LS258	1.25 *

Order Minimum $10.00. Add $1.00 shipping and handling charge per order. California residents add **6.5**% *sales tax. All orders shipped First Class within 24 hours.*

Order the famous Iasis 6 volume Programmed Learning Course "Microcomputer Design is a Snap" for $99.50 and receive a special $10.00 credit on any group of IC's.

Satisfaction 100% guaranteed.

C.O.D. Orders: Phone (day or night) 408/354-1448

ELTRON

PO BOX 2542B
Sunnyvale, CA 94087

Free catalog—Just send us your name and address

---

cybercom BOARDS

MB-1 MK-8 Computer RAM, (not S-100), 4KX8, uses 2102 type RAMs, PCBD only $22

MB-3 1702A EROM Board, 4KX8, S-100, switchable address and wait cycles, kit less PROMS $65

MB-4 Basic 4KX8 ram, uses 2102 type rams, may be expanded to 8KX8 with piggybacking, S-100 buss. PC board $30

MB-6 Basic 8KX8 ram uses 2102 type rams, memory protect in 256 to 8K switchable S-100 buss. PCBD $35

MB-8 2708 EROM board, S-100, 8KX8 or 16KX8 kit without PROMS $85

IO-2 S-100, 8 bit parallel I/Oport, ⅔ of board is for kludging. Kit ...... $55 PCBD $30

VB-1 64X 16 video board, upper lower case Greek, composite and parallel video with software, S-100. Kit ...... $189.00 PCBD $35

SP-1 Music synthesizer board, S-100, computer controller wave forms, 9 octaves, 1V rms ½% distortion, includes software kit $250

Altair Compatible Mother Board, 11 x 11½ x ⅛".
Board only ... $45 With 15 connectors $105

Extender Board full size. Board only $9
With connector $13.50

Solid state music Cybercom boards are high quality glass board with gold finger contacts. All boards are check for shorts. Kits only have solder mask. 90 day guarantee on Cybercom kits.

Waneco Boards

MM-1 8KX8 fully buffered, S-100, uses 2102 type rams. PCBD $30

Mother Board 12 slot, terminated, S-100, board only $40

Special 8080A processor & support chip set kit includes one of each; all parts prime 9080A (8080A); 8228; 8212; 8216; 8251; 8214; 8224; 8225. Special $75.00

Special 2101AL-4 1K x 1 ram ⅓ less power than 21L02 type rams, with power down, prime from NEC. Ea. 2.00; 32 ea. 1.80; 64 ea. 1.70; 128 ea. 1.60; 256 ea. 1.50; 512 ea. 1.45.

9080A AMD 8080A (Prime)	25.00
8212/74S412 Prime	4.00
8214 Prime	8.30
8216 Prime	4.95
8224 Prime	5.00
8228 Prime	8.90
8251 Prime	14.50
8255 Prime	14.50
1702A-6 AMD 4702A Prime	8.00
TMS-6011 UART Prime	6.95
2513 Char Gen Upper Prime	11.00
2513 Char Gen Lower Prime	11.00
1702A Intel Not Prime	8.00

8T10	2.00	8T97	2.00	80L97	1.50
8T13	2.50	8T110	2.00	81L22	1.50
8T16	2.00	5309	8.00	82L23	1.90
8T20	2.00	5312	4.00	85L51	2.50
8T24	2.50	5313	4.00	85L52	2.50
8T26	2.75	5320	5.95	85L63	1.25
8T34	2.50	5554	1.90	86L70	1.50
8T37	2.50	5556	2.50	86L75	1.90
8T38	2.50	5055	1.60	86L99	3.50
8T80	2.50	MC4044	2.25	88L12	.80

MIKOS

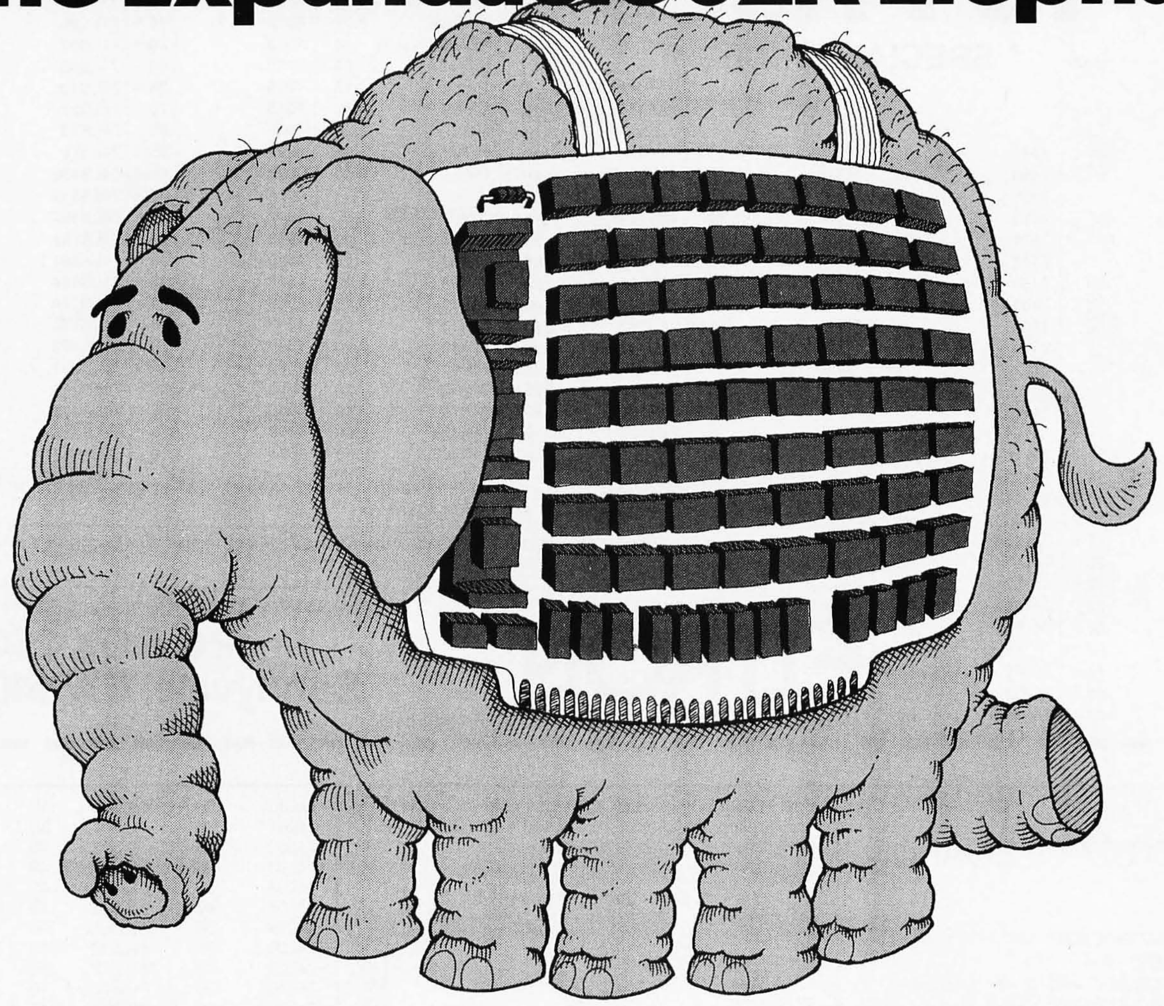
419 Portofino Drive
San Carlos, California 94070
Please send for IC, Xistor and Computer parts list.

74L00	.25	74LS00	.40	1101	1.25
74L01	.25	74LS01	.50	1103	1.25
74L02	.25	74LS02	.40	2101	4.50
74L03	.25	74LS03	.40	2111-1	3.75
74L04	.30	74LS04	.45	2112	4.50
74L05	.40	74LS05	.45	2602	1.60
74L06	.30	74LS08	.40	4002-1	7.50
74L08	.40	74LS10	.40	4002-2	7.50
74L09	.40	74LS12	.55	MM5262	1.00
74L10	.30	74LS20	.40	7489	2.00
74L20	.35	74LS22	.45	74200	4.95
74L26	.40	74LS27	.45	74C89	3.00
74L30	.40	74LS30	.40	82S06	2.00
74L32	.45	74LS37	.60	82S07	2.00
74L42	1.50	74LS38	.60	82S17	2.00
74L51	.35	74LS42	1.50	8223	2.50
74L54	.45	74LS51	.40	82S23	3.00
74L55	.35	74LS54	.45	82S123	3.00
74L71	.30	74LS55	.40	82S126	3.50
74L73	.55	74LS73	.65	82S129	3.50
74L74	.55	74LS74	.65	82S130	3.95
74L75	1.20	74LS76	.65	82S131	3.95
74L78	.90	74LS151	1.55	IM5600	2.50
74L85	1.40	74LS174	2.20	IM5610	2.50
74L86	.75	74LS175	1.95	IM5603	3.00
74L89	3.50	74LS192	2.85	IM5604	3.50
74L90	1.50	2501B	1.25	IM5623	3.00
74L91	1.50	2502B	3.00	IM5624	3.50
74L93	1.70	2507V	1.25	MMI6330	2.50
74L95	1.70	2510A	2.00	DM8573	4.50
74L98	2.80	2517V	1.25	DM8574	5.50
74L123	1.50	2519B	2.80	DM8575	4.50
74L164	2.50	2532B	2.80	DM8576	4.50
74L165	2.50	2533V	2.80	DM8577	3.50
74L192	1.25	DM8131	2.50	DM8578	4.00
74L193	1.20	N8263	3.50	2.4576 MHZ	
MH0026	2.95	MC1489	1.50	XTAL	7.20
MC1488	1.50	DM8837	1.50		

Check or money order only. If you are not a regular customer and your order is large please send either a cashier's check or a postal money order, otherwise there will be a delay of two weeks for the check to clear. All items post paid in the U.S. Calif. residents add 6% tax. Money back 30 day guarantee. We cannot accept returned IC's that have been soldered to. Prices subject to change without notice. $10 minimum order.

Circle 190 on inquiry card.

ARTEC Introduces
The Expandable 32K Elephant

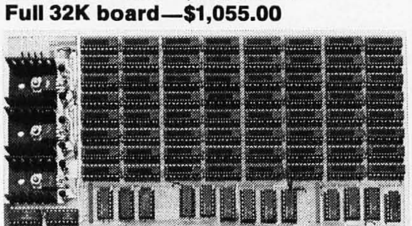


The 8K-32K Expandable Memory That Grows With Your System

Now, for the first time, you can have a reliable true static memory that will grow with your system. Start with the board and 8K memory. Then add on one, two or three 8K increments of memory up to 32K. 250 ns access time. The Artec 32K Expandable Memory allows you plenty of room for memory and all necessary support hardware.

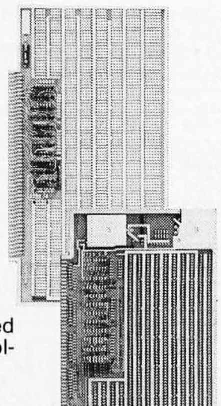
For five years Artec craftsmanship and reliability has been proven in tough industrial use. Now, you too can enjoy breadboards and memories that will work time after time. Boards like the GP 100 and the wire wrap WW-100. Send for an Artec Board, your order will be sent the same day as received.

Board & 8K of memory—$290.00
8K add on kits—$255.00 ea.
Full 32K board—$1,055.00



GP-100—$20.00
Maximum design versatility along with standard address decoding and buffering for S—100 systems. Room for 32 uncommitted 16 pin IC's, 5 bus buffer & decoding chips, 1 DIP address select switch, a 5 volt regulator and more. High quality FR4 epoxy. All holes plated through. Reflowed solder circuitry.

WW-100—$20.00
A wire wrap breadboard, similar to the GP 100. Allows wirewrap of all sizes of sockets in any combination. An extra regulator position for multiple voltage applications. Contact finger pads arranged for easy pin insertion.



TO ORDER: Use your Mastercharge or BankAmericard. Or just send along a money order. Your order will get same day service.

FOR MORE INFORMATION: For more information about these or any of Artec's complete line of circuit boards or for either industrial or personal use, please call or write. A catalogue will gladly be sent.

Please send me:
☐ 32K ☐ GP-100 ☐ WW-100
☐ I've enclosed a money order.
Bill my ☐ Mastercharge
☐ BankAmericard No. ________
Name ________
Address ________
City ________ State ________ Zip ________

10% discount for students & computer club members.

ARTEC ELECTRONICS, INC.

605 Old County Rd. • San Carlos, CA 94070 • (415) 592-2740

Circle 195 on inquiry card.

Circle 191 on inquiry card.

S.D. SALES CO. P.O. BOX 28810 - B DALLAS, TEXAS 75228

Z-80 CPU KIT
For Imsai-Altair
$149. kit

Z-80 Chip & Manual $49.95

From the same people who brought you the $89.95 4K RAM Kit. We were not the first to introduce an Imsai/Altair compatible Z-80 card, but we do feel that ours has the best design and quality at the lowest price!

The advanced features of the Z-80 such as an expanded set of 158 instructions, 8080A software compatibility, and operation from a single 5VDC supply, are all well known. What makes our card different is the extra care we took in the hardware design. The CPU card will always stop on an M1 state. We also generate TRUE SYNC on card, to insure that the rest of your system functions properly. Dynamic memory refresh and NMI are brought out for your use. Believe it or not, not all of our competitors have gone to the extra trouble of doing this.

As always this kit includes all parts, all sockets, and complete instructions for ease of assembly. Because of our past experience with our 4K kit we suggest that you order early. All orders will be shipped on a strict first come basis. Dealers inquiries welcome on this item. Kit includes Zilog Manual and all parts. Kit shipped with 2 MHZ crystals.

Z–80 MANUAL – $7.50 SEPARATELY

THE WHOLE WORKS
$89.95

4K LOW POWER RAM BOARD KIT

Imsai and Altair 8080 plug in compatible. Uses low power static 21L02–1 500 ns. RAM'S. Fully buffered, drastically reduced power consumption, on board regulated, all sockets and parts included. Premium quality plated through PC Board.

For 250 ns RAM's add $10.00

Music to your Ears!

CAR/BOAT KIT
$34.95

NEWEST KIT FROM S.D. SALES!

MUSICAL HORN

HOME KIT
$19.95

Musical Horn Kit for Car, Boat or Home
Plays any tune from Mozart to Led Zeplin
Change tunes in seconds
Complete Solid State electronics
Standard or custom tunes available at $6.95 each (you supply us with the sheet music – we supply electronics for your favorite tunes.)
One song supplied with original order

Standard Tunes Available:
DIXIE – EYES OF TEXAS – ON WISCONSIN – YANKEE DOODLE DANDY – NOTRE DAME FIGHT SONG – PINK PANTHER – AGGIE WAR SONG – ANCHORS AWAY – NEVER ON SUNDAY BRIDGE OVER RIVER QUI – CANDY MAN

Home Kit includes speaker which operates from your door bell. When door bell is pushed your favorite tune is played. Car/Boat Kit DOES NOT include speaker. Uses standard 8ohm PM speaker. Allow 4 weeks delivery on both kits.

Limited Quantity!

$9.95 kit

SIX DIGIT ALARM CLOCK KIT

We made a fantastic kit even better. Redesigned to take advantage of the latest advances in I.C. clock technology. Features: Litronix Dual ½" displays. Mostek 50250 super clock chip, single I.C. segment driver, SCR digit drivers. Greatly simplified construction. More reliable and easier to build. Kit includes all necessary parts (except case). For P.C.Board add $3.00; AC XFMR add $1.50. Do not confuse with Non-Alarm kits sold by our competition! Eliminate the hassle – avoid the 5314!

NEW! WITH JUMBO LED READOUTS!

1000 MFD Filter Caps Rated 35 WVDC Upright style with PC leads. Most popular value for hobbyists.	SLIDE SWITCH Assortment Our best seller. Includes miniature and standard sizes, single and multi-position units. All new.	POWER RESISTOR 15 OHM 25W BY CLAROSTAT	RESISTOR ASSORTMENT ¼W 5% & 10% PC leads. A good mix of values! Special!	P.C. LEAD DIODES 1N4148/1N914 100/$2.00 1N4002 - 1A 100 PIV	Just received a good mixed lot of National TO-92 plastic transistors. PNP & NPN, even a few FET's. 40-50% yield. Untested Asst.	DISC CAP ASSORTMENT P.C. Leads. At least 10 different values. Includes .001, .01, .05 plus other standard values.
4/$1.00	12/$1.00	75¢ ea.	200/$2.	40/$1.	500/$3.	60/$1.00

AMD-1702A

Huge Factory Purchase

FACTORY PRIME UNITS! BRAND NEW!
1.5 Micro-Seconds Access Time.

10/$40. $4.95 ea.

IC's from XEROX

7400 – 9c	7430 – 9c	7493 –26c
7402 – 9c	7440 – 9c	74121 –22c
7404 – 9c	7437 –10c	74123 –32c
7406 –11c	7438 –10c	74151 – 9c
7407 –11c	7451 – 9c	74155 –22c
7410 – 9c	7474 –16c	74193 –35c
7416 –13c	7475 –24c	8233 – 35c
7420 – 9c	7486 –16c	Intel – 1302 – 45c

1402 A Shift Regulator – 50c
MH0025CN – 55c

IC's REMOVED FROM PC BOARDS ALL TESTED; FULL SPEC.

UP YOUR COMPUTER!
21L02-1

And so is power! Not only are our RAM'S faster than a speeding bullet but they are now very low power. We are pleased to offer prime new 21L02-1 Low Power and Super Fast RAM's. Allows you to STRETCH your power supply farther and at the same time keep the wait light off!

500 ns 8/$12.95
250 ns 8/$15.95

$12.95 *S.D. Sales Exclusive!* $12.95

MOS 6 DIGIT UP/DOWN COUNTER

40 PIN DIP. Everything you ever wanted in a counter chip. Features: Direct LED sement drive, single power supply (12 VDC TYPE.), six decades up/down, pre-loadable counter, separate pre-loadable compare register with compare out-put, BCD and seven segment outputs, internal scan oscillator, CMOS compatible, leading zero blanking. 1MHZ. count input frequency.

VERY LIMITED QUANTITY!

Special!

3.579545 MHZ Time Base Crystal $1.25

28 PIN IC Sockets 3/$1.00

39 MFD 16V Mallory Electrolytic 15/$1.00

11,000 MFD 50WVDC Computer Grade Cap. $3.00

TERMS:

MONEY BACK GUARANTEE!

No COD's. Texas Residents add 5% Sales Tax. Add 5% of order for postage & handling. Orders under $10. add 75c. Foreign orders: U.S. Funds Only!

Call in your BANKAMERICARD or MASTER CHARGE order in on our Continental United States Toll Free Watts Line:

1-800-527-3460

Texas Residents Call Collect:

214/271-0022

Cheap DC Supply

Actually an AC adaptor for calculators. 9VDC no load. 6VDC @200ma. 4VDC @ 375ma.

$2. ea.

S. D. SALES CO.
P.O. BOX 28810 – B
DALLAS, TEXAS 75228

Orders over $15. - Choose $1. FREE MERCHANDISE!

Circle 192 on inquiry card.

Reader Service

To get futher information on the products advertised in BYTE, fill out the reader service card with your name and address. Then circle the appropriate numbers for the advertisers you select from this list. Add a 9 cent stamp to the card, then drop it in the mail. Not only do you gain information, but our advertisers are encouraged to use the marketplace provided by BYTE. This helps us bring you a bigger BYTE.

***Reader service inquiries not solicited. Correspond directly with company.**

BOMB—BYTE's Ongoing Monitor Box

Defused

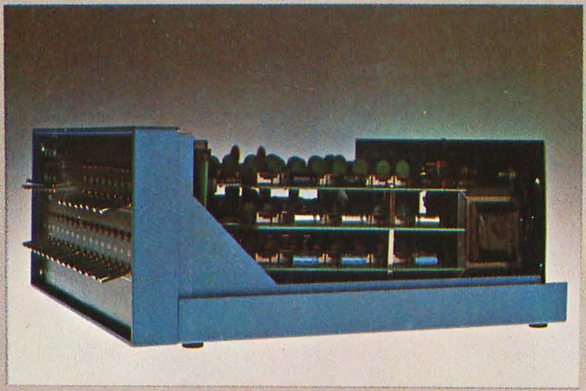
This month, there is no BOMB analysis for May BYTE. The BOMB got defused by the late delivery of the May 1977 isssue due to the strike at our printer. As a result, we are holding the analysis until next month when a double BOMB will report the results of May and June reader response together.■

Powerful in computing muscle, yet small in physical size, the Altair™680b offers many special features at an affordable price. Based on the 6800 microprocessor, the 680b comes with 1K of static RAM, Serial I/O port, PROM monitor and provisions for 1K of PROM as standard components. It's good thinking, when you're interested in making a modest investment on a highly reliable computer, to consider the Altair 680b.

Our PROM monitor eliminates the necessity for toggling front panel switches to load bootstraps or manipulate memory contents. Only a terminal and programming language are required for complete system operation. With Altair System software—Altair 680 BASIC, assembler and text editor—you may begin problem solving immediately with ease.

By adding the 680b-MB Expander card, many options are currently available:

*16K Static Memory Board—Increase your system memory with 16K bytes of fast access (215 ns), low power (5 watts per board) static RAM. 680 BASIC and assembler/text editor are included free with purchase.



*Process Control Interface—A PC card that uses optically isolated inputs and relay outputs that transmit sensory information to and control signals from the computer. A diverse world of control applications is opened up with the Altair 680b-PCI.

*Universal Input/Output Board—If your I/O needs exceed the serial port already on the main board, augment your I/O channels with the 680b-UI/O. By implementing the optional serial port and two parallel ports, you can simultaneously interface to four terminals.

*New Addition—Kansas City Audio Cassette Interface—Use the 680b-KCACR to interface your Altair 680b with an audio cassette recorder for inexpensive mass storage of programming languages, programs and data.

Available in either full front panel or turnkey models, the Altair 680b presents many computing capabilities at a low cost—without skimping on performance. See it today at your local Altair Computer Center or contact the factory for further details.

mits

Good Thinking.



Circle 146 on inquiry card.

2450 Alamo S.E. Albuquerque, New Mexico 87106
dealer inquiries invited.